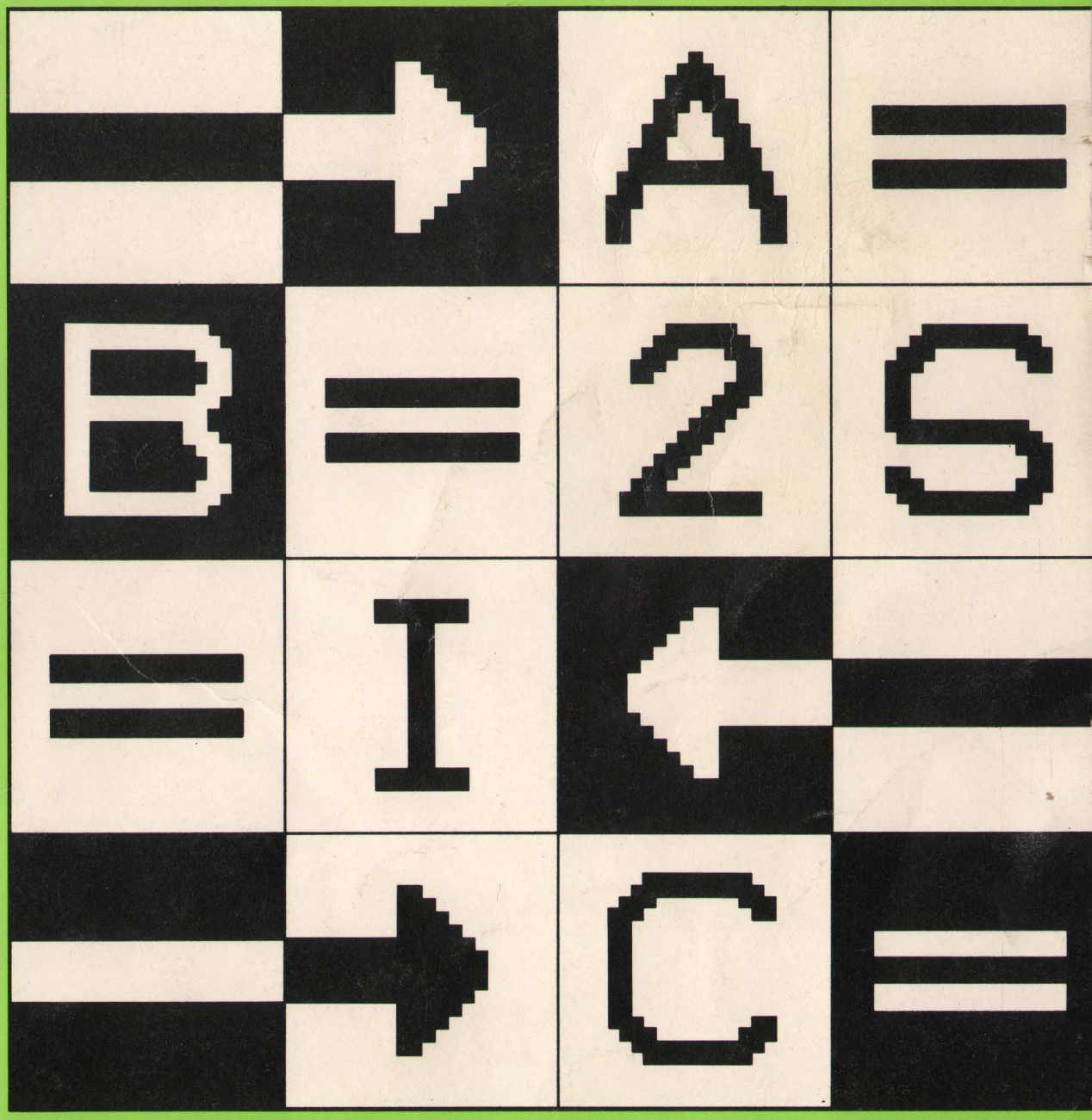
Texas Instruments TI-99/4A Computer

Bedienungsanleitung



WICHTIGE ANMERKUNG PROGRAMME UND BUCHMATERIAL

BEZÜGLICH DIESER PROGRAMME ODER BUCHMATERIALIEN ODER ALLER DAVON ABGELEITETEN PROGRAMME GEWÄHRT TEXAS INSTRUMENTS KEINE AUSDRÜCKLICHE ODER STILLSCHWEIGENDE GARANTIE – EINSCHLIESSLICH, ABER NICHT BESCHRÄNKT AUF ETWAIGE STILLSCHWEIGENDE GARANTIEN DER HANDELSFÄHIGKEIT UND EIGNUNG FÜR EINEN BESTIMMTEN ZWECK – UND STELLT DIESE MATERIALIEN NUR AUF DER GRUNDLAGE DESSEN, WAS IST, ZUR VERFÜGUNG.

TEXAS INSTRUMENTS ÜBERNIMMT KEINESFALLS DIE HAFTUNG FÜR BESONDERE, ZUSÄTZLICHE, NEBEN- ODER FOLGESCHÄDEN, DIE IN ZUSAMMENHANG MIT ODER DURCH DEN KAUF VON, BEZIEHUNGSWEISE DIE VERWENDUNG DIESER BUCHMATERIALIEN ODER PROGRAMME ENTSTEHEN, UND DIE EINZIGE UND AUSSCHLIESSLICHE VERBINDLICHKEIT FÜR TEXAS INSTRUMENTS ÜBERSCHREITET NICHT DEN KAUFPREIS DIESES BUCHES, UNABHÄNGIG VON DER ART DER ERGRIFFENEN MASSNAHME. WEITERHIN GEHT TEXAS INSTRUMENTS KEINE VERPFLICHTUNGEN FÜR ETWAIGE FORDERUNGEN JEGLICHER ART EIN, DIE GEGEN DEN BENUTZER DIESER PROGRAMME ODER BUCHMATERIALIEN DURCH DRITTE ERHOBEN WERDEN.

ISBN/0-89512-048-8

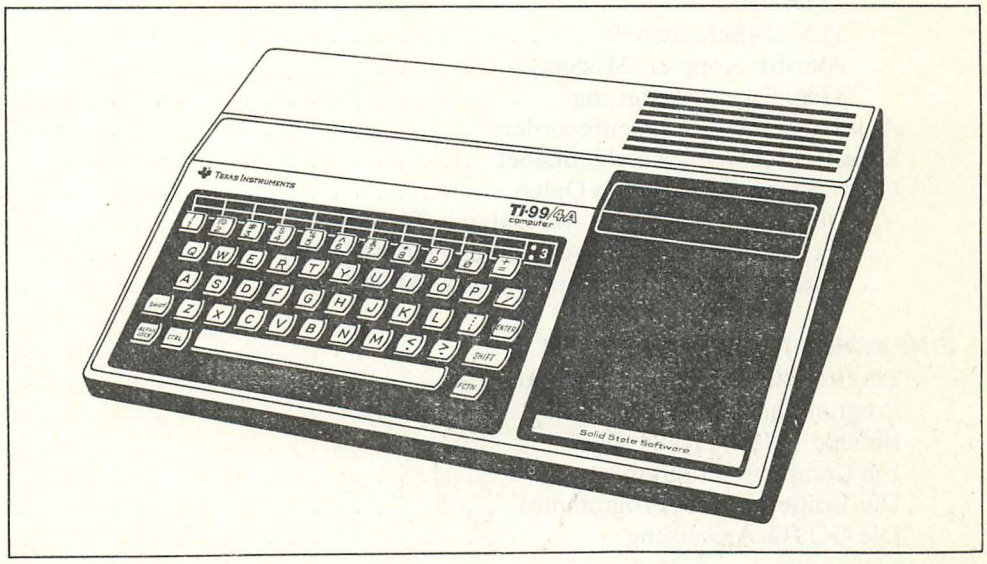
© Copyright 1982 by Texas Instruments Inc.

TEXAS INSTRUMENTS TI-99/4A HOME-COMPUTER SYSTEM

LCB-4491

Bedienungsanleitung

Ihr Begleiter auf dem Weg zum Programmieren
mit dem Texas Instruments TI-99/4A Home-Computer



Bitte lesen Sie zuerst die beiligende
"Bedienungsanleitung Hardware" aufmerksam durch

Inhalt

I. ALLGEMEINE INFORMATIONEN

Auf den Spuren in eine neue Welt . . . 6
Praktisches Modulsystem . . . 6
Diskettenprogramme . . . 6
Leistungsfähige Programmiersprachen . . . 6
Die Anwendung dieses Buchs . . . 6
Eine Übersicht zu Ihrem Computer . . . 7
Das Tastenfeld . . . 7
Dauerfunktion . . . 7
Feststelltaste für Großschreibung . . . 7
Interpunktions- und Symbol-Tasten . . . 7
Sonderfunktionstasten . . . 7
Spezielle Steuertasten . . . 8
Tastatur-Schablone . . . 8
Mathematik- oder Operationstasten . . . 9
Leertaste . . . 9
Fehlerkorrektur . . . 9
Systemerweiterung . . . 10
TI Fernbedienung . . . 10
TI Sprachsynthesizer . . . 10
TI Disketten-System . . . 10
TI Drucker . . . 10
TI V. 24 Schnittstelle . . . 10
Akustik-Koppler (Modem) . . . 10
TI Speichererweiterung . . . 10
Anschluß eines Kassettenrecorders . . . 11
Kassettenrecorder-Anschlußkabel . . . 11
Übertragung/Eingabe von Daten . . . 12
Übertragung/Eingabe von Daten in TI-BASIC . . . 12
Datenübertragung bei Anwendung eines Moduls . . . 12
Dateneingabe bei Anwendung eines Moduls . . . 14

II. EINFACHES PROGRAMMIEREN . . . 15
Ein Programm für das Ausdrucken . . . 16
Programmaufbau . . . 17
Befehle – NEW, RUN, LIST . . . 18
Ein Computer-Programm . . . 19
Das Editieren eines Programmes . . . 21
Die GOTO-Anweisung . . . 28

Inhalt

III. TI-BASIC BEFEHLE UND STATEMENTS

Einführung . . . 30
Gliederung dieser Anleitung . . . 30
Vereinbarungen zur Schreibweise . . . 30
Beispiele . . . 30
Allgemeine Informationen . . . 31
Einführung . . . 31
Spezielle Tastenfunktionen . . . 31
Leerstellen . . . 32
Zeilennummern . . . 33
Numerische Konstanten . . . 34
Exponentialform . . . 34
Bereichsunterschreitung . . . 34
Kapazitätsüberlauf . . . 34
String-Konstante . . . 35
String-Ausdrücke . . . 35
Variable . . . 36
Reservierte Wörter . . . 37
Numerische Ausdrücke . . . 38
Vergleiche und logische Variable . . . 39
Commands . . . 40
Einführung . . . 40
NEW . . . 40
LIST . . . 40
RUN . . . 42
BYE . . . 42
NUMBER . . . 43
Editieren im Number-Modus . . . 44
RESEQUENCE . . . 45
BREAK . . . 45
CONTINUE . . . 47
UNBREAK . . . 47
TRACE . . . 49
UNTRACE . . . 49
EDIT . . . 50
SAVE . . . 51
OLD . . . 53
DELETE . . . 54
Allgemeine Programmbefehle (Statements) . . . 55
Einführung . . . 55
LET . . . 55
REM . . . 56
END . . . 57
STOP . . . 57
GOTO . . . 57
ON-GOTO . . . 58
IF-THEN-ELSE . . . 58
FOR-TO-STEP . . . 60
NEXT . . . 63
INPUT-OUTPUT-Statements (Ein-/Ausgabeanweisungen) . . . 64
Einführung . . . 64
INPUT . . . 64
READ . . . 67
DATA . . . 68
RESTORE . . . 69

Inhalt

PRINT 70
DISPLAY 73
Farbgraphiken und akustische Signale 73
Einführung 73
CALL CLEAR 74
CALL HCHAR 75
CALL VCHAR 76
CALL GCHAR 76
CALL COLOR 77
CALL SCREEN 79
CALL CHAR 80
CALL SOUND 84
CALL KEY 87
CALL JOYST 91
Eingebaute numerische Funktionen 92
Einführung 92
ABS 92
ATN 93
COS 93
EXP 93
INT 94
LOG 94
RANDOMIZE 95
RND 95
SGN 96
SIN 96
SQR 97
TAN 97
Eingebaute String-Funktionen 98
Einführung 98
ASC 98
ASCII-Zeichenkodes 99
CHR$ 100
LEN 100
POS 101
SEG$ 101
STR$ 102
VAL 102
Anwenderdefinierte Funktionen 103
Einführung 103
DEF 103
Datenfelder 106
Einführung 106
DIM 107
OPTION BASE 109
Unterprogramme (Subroutinen) 109
Einführung 109
GOSUB 110
RETURN 112
ON-GOSUB 113

Inhalt

Dateiverarbeitung 114
Einführung 114
OPEN 115
CLOSE 119
INPUT 121
EOF 126
PRINT 127
RESTORE 131

IV. FEHLERMELDUNGEN 132

V. ANWENDERPROGRAMME 137

Die Raupe 137
Zeltdach 138
Geheimzahl 139
Ballspiel 141
Kontostand überprüfen 143
Graphikkombinationen 145

VI. BEGRIFFE UND ERKLÄRUNGEN 152

Allgemeines

Auf den Spuren in eine neue Welt ...

Hilfe bei der Erziehung Ihrer Kinder, bei der Aus- und Weiterbildung im Erwachsenenalter, kreative Unterhaltung für die ganze Familie, Arbeitserleichterung im Privatleben und Beruf und beim beruflichen Fortkommen – eine völlig neue Welt des Lernens! Die erschließt Ihnen der Home-Computer TI 99/4A von Texas Instruments. Vergessen Sie alles, was Sie von Computern wissen oder meinen. Texas Instruments bringt Ihnen moderne Computerleistung ins Haus. Folgen Sie den Spuren in diese neue Welt!

Dieses Buch stellt Ihnen die aufregende, neue Welt der Computer vor.

Der Home-Computer TI 99/4A hilft zum Beispiel beim Erlernen einer Sprache und bei der Mathematik (sei es für die Schule oder für den Beruf), ebenso wie allgemein beim Gebrauch und Programmieren von Computern.

Spannende und lehrreiche (logisch-strategische) Spielprogramme und lehrreiche Musikprogramme sorgen für eine kreative Freizeitgestaltung zu Hause. Auch bei der Haushaltsführung ist der TI 99/4A behilflich: er sorgt für eine sinnvolle Planung des Haushaltsgeldes.

Mit dem TI 99/4A wird Ihr Fernsehgerät zur aktiven Kommunikations-"Drehscheibe". Sie brauchen keine Vorkenntnisse. Die einfachen Anweisungen in diesem Buch, die Beschreibungen zu den Programm-Modulen und Bildschirm-Dialog – das ist alles!

Praktisches Modulsystem

Das einzigartige System der steckbaren Solid State Software*-Programm-Module macht Ihnen die Sache zum Vergnügen. Diese Module sind für Sie vorprogrammiert: Einstecken und anfangen. Schritt für Schritt werden Sie durch die einzelnen Programme geführt. Sie haben eine reiche Module-Auswahl. Informationen finden Sie bei Ihrem Händler oder bei Texas Instruments.

Diskettenprogramme

Neben den Programm-Modulen gibt es ein umfangreiches Spektrum an Software auf Disketten.
Wie die Programm-Module ist diese Software sofort einsatzbereit, ohne daß Sie selbst programmieren müssen. Informieren Sie sich bei Ihrem Händler.

Leistungsfähige Programmiersprachen

- TI-BASIC, eine einfache, aber sehr leistungsfähige Computersprache, ist fest in den Home-Computer eingebaut. Mit TI-BASIC können Sie Ihre eigenen Programme entwickeln, zusätzlich mit Farbgrafik über Tonfolgen bis zur gesprochenen Sprache. Darüber hinaus können Sie mit anderen Programmiersprachen arbeiten.
- TI LOGO ist eine leichtverständliche Computer-Sprache, die Schülern jeder Altersstufe die Kommunikation mit dem Computer ermöglicht. TI LOGO schafft die computer-bezogene Voraussetzung, mit der z. B. Mathematik und anderes "formales" Lernen in natürlicher Sprache abläuft.
- TI EXTENDED BASIC ist eine leistungsfähige Programmier-Hochsprache mit einer Vielzahl von Funktionen, die in TI-BASIC nicht gegeben sind. Kurze Hinweise auf die Möglichkeiten finden Sie im BASIC-Teil.
- UCSD PASCAL ist eine sorgfältige durchstrukturierte, leistungsfähige Programmiersprache, die schneller, logischer und noch wirkungsvoller arbeitet als BASIC. Sie ist bestens geeignet für die Ausbildung zum Programmierer und bietet dem Fortgeschrittenen mehr Programmiermöglichkeit. PASCAL-Programme sind block-strukturiert, so daß logische Elemente eines Programms jeweils als eine Einheit entwickelt werden können.
- Mit Hilfe von TMS 9900 EDITOR/ASSEMBLER schreiben Sie Programme in der Sprache des Mikroprozessors, der in den Home-Computer eingebaut ist. Der Programmieraufwand ist zwar größer als bei BASIC, TI-LOGO oder UCSD PASCAL, dafür sind Sie aber in der Lage, in der Ausführung sehr schnelle und bis ins kleinste Detail des Systems greifende Programme zu entwickeln.

Die Anwendung dieses Buches

Diese Bedienungsanleitung ist wie folgt gegliedert:

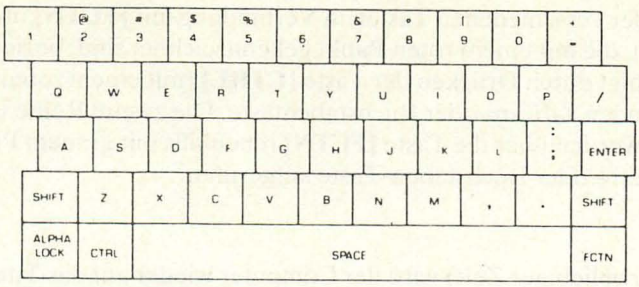
- Allgemeiner Überblick über Anschlüsse und Tastatur
- Systemerweiterung
- Einführung in das Programmieren
- Ausführliche Erläuterung von TI-BASIC
- Anwendungsbeispiele
- Begriffsbestimmungen

Und nun: Alles Gute für Sie auf den Spuren in die Neue Welt des Lernens – mit dem Home-Computer TI 99/4A von Texas Instruments.

* US Trade Mark von Texas Instruments ** Separat als Zubehör

Tastenfeld

Das Tastenfeld



Der TI 99/4A hat eine Standard-Schreibmaschinen-Tastatur mit verschiedenen Tastentypen. Drückt man bei TI-BASIC eine beliebige Taste, wird ein Buchstabe in Kleinschreibung (kleiner Großbuchstabe) bzw. das untere Zeichen der Taste auf dem Bildschirm angezeigt.

Drückt man SHIFT und gleichzeitig eine andere Taste, erhält man in der Anzeige Großschreibung oder das entsprechende obere Zeichen der Taste. Mit Ausnahme der Buchstabentasten sind Sonderzeichen, die mit Hilfe der Taste SHIFT geschrieben werden, jeweils über dem Symbol angegeben, das man bei Kleinschreibung erhält.

Einige Tasten haben zusätzliche Sonderfunktionen, die in den folgenden Abschnitten erklärt werden.

Dauerfunktion

TI-BASIC ist mit einer automatischen Dauerfunktion ausgestattet. Drückt man die Leertaste oder eine beliebige andere Taste länger als eine Sekunde, wird das Zeichen solange wiederholt, bis Sie die Taste loslassen.

Feststelltaste für Großschreibung

Die dauernde Großschreibung bei allen Buchstaben-Tasten erreicht man durch Einrasten der Taste **ALPHA LOCK.** Die Ziffern- und Interpunktions-Tasten werden damit nicht beeinflußt. Durch erneutes Drücken von **ALPHA LOCK** wird die Großschreibung wieder aufgehoben.

Achtung

Mit Ihrem Computer können Sie die Zahl "1" nicht als "L" eintippen. Ebenso dürfen Sie eine Null niemals durch den Buchstaben "O" ersetzen.

Zur besseren Unterscheidung zeigt der Computerbildschirm den Buchstaben "O" in rechteckiger Form, die Zahl "0" in runder Form an.

Interpunktions- und Symbol-Tasten

Die Computer-Tastatur verfügt neben den Interpunktions- und Symbol-Tasten, die auf einer Standard-Schreibmaschine erscheinen, noch über einige andere, die man in Computeranwendungen benutzt. Um ein Symbol zu tippen, das auf der unteren Tastenhälfte angegeben ist, drücken Sie einfach die gewünschte Taste. Um das Symbol darüber (Großschreibungs-Symbol) zu tippen, drücken Sie **[SHIFT]** und gleichzeitig die gewünschte Taste. Beachten Sie, daß Interpunktionszeichen und Symbole auch an einigen Tasten vorne stehen. Wenn Sie diese Symbole tippen wollen, drücken Sie **[FCTN]** und die erforderliche Taste.

Sonderfunktionstasten

Mehrere Tasten haben verschiedene Funktionen in TI-BASIC, in der Modul-Software und in anderen Anwendungsbereichen.

Der genaue Zweck dieser Tasten ist in den entsprechenden Abschnitten dieses Buches oder in den Beschreibungen zu den verschiedenen Moduln detailliert beschrieben.

Um eine Sonderfunktion zu aktivieren (ausgenommen ENTER), drücken Sie **[FCTN]** und gleichzeitig die erforderliche Taste.

Sonderfunktionen

Tastatur-Schablone

Eine Zweifach-Tastatur-Schablone ist im Lieferumfang Ihres Computers inbegriffen. Sie können diese Schablone verwenden, um die Funktionen der verschiedenen Tasten in Verbindung mit **[FCTN]** und **[CTRL]** besser zu erkennen. Die obere Reihe von Funktionen, die mit einem roten Punkt gekennzeichnet sind, bezieht sich auf die Steuertasten. Der Zugriff auf diese Funktionen erfolgt durch Drücken der Taste **[CTRL]** (mit einem roten Punkt markiert) und gleichzeitiges Drücken der entsprechenden Ziffern- oder Buchstabentaste. Die zweite Reihe von Funktionen, die mit einem hellgrauen Punkt markiert sind, werden über die Taste **[FCTN]** (ebenfalls mit grauem Punkt) und gleichzeitiges Drücken der erforderlichen Ziffern- oder Buchstaben-Taste aufgerufen.

[FCTN =] (QUIT)

Durch Drücken von QUIT (zu beliebiger Zeit) wird der Computer wieder auf das Titelbild geschaltet.
Anmerkung: Wenn Sie QUIT drücken, werden eingegebene Daten oder Programme gelöscht.

[ENTER] (Eingabe)

In den meisten Fällen gilt: wenn Sie die Taste **[ENTER]** drücken, weisen Sie den Computer an, die soeben eingetippten Informationen zu übernehmen. Weitere Funktionen werden in den entsprechenden Beschreibungen erklärt.

[FCTN ←] (LINKS)

Beim Drücken der Taste mit dem linksgerichteten Pfeil (Rücktaste) wird der Positionsanzeiger nach links gerückt. Die Zeichen auf dem Bildschirm werden nicht gelöscht oder geändert, wenn der Positionsanzeiger darüber hinwegläuft.

[FCTN →] (RECHTS)

Beim Drücken der Taste mit dem rechtsgerichteten Pfeil (Vorwärtstaste) wird der Positionsanzeiger nach rechts gerückt. Wenn der Positionsanzeiger gedruckte Zeichen auf dem Schirm passiert, werden sie in keiner Weise geändert.

[FCTN ↑] (UP)
[FCTN ↓] (DOWN)

Diese Tasten haben je nach Anwendungsbereich verschiedene Funktionen. Vollständige Informationen über ihre Anwendung finden Sie im Abschnitt TI-BASIC und in der Dokumentation zur jeweiligen Software.

[FCTN 2] (INS)

Die "Insert"-Taste wird verwendet, um in die Zeilen, die Sie tippen, noch einen Buchstaben, eine Zahl oder ein anderes Zeichen einzufügen.

[FCTN 1] (DEL)

Die "Delete"-Taste verwendet man zum Löschen eines Buchstabens, einer Zahl oder eines Zeichens aus der Zeile, die Sie tippen.

[FCTN 3] (ERASE)

Drückt man die **[ERASE]**-Taste, wird die Zeile, die Sie gerade tippen, gelöscht.

[FCTN 4] (CLEAR)

Diese Taste verwendet man im allgemeinen, um eine beliebige eingetippte Information (vor Drücken der Taste **[ENTER]**) zu löschen. Sie hat zusätzliche Funktionen in TI-BASIC (siehe "Spezielle Tastenfunktionen" im BASIC-Teil dieser Anleitung). Andere Tasten haben Sonderfunktionen bei der Software.

Einige davon sind:
[FCNT 5] (BEGIN) **[FCNT 6]** (PROC'D) **[FCNT 7]** (AID) **[FCNT 8]** (REDO) **[FCNT 9]** (BACK)

Spezielle Steuertasten

Ihr TI Computer verfügt über Steuertasten, die primär der Daten(fern)übertragung dienen. Eine Liste der Standard ASCII-Steuerzeichen finden Sie im Anhang unter " Kodes für Steuertasten".

Um ein Steuerzeichen einzugeben, drücken Sie die CTRL-Taste und gleichzeitig die erforderliche Buchstaben- oder Zifferntaste.

Sonderfunktionen

Mathematik- oder Operationstasten

Mit den Mathematik- (oder Operations-) Tasten wird der Computer angewiesen, Addition, Subtraktion, Multiplikation und Division durchzuführen oder eine Zahl zu potenzieren.

Die Symbole für Addition, Subtraktion und das Gleichheitszeichen kennen Sie bereits, aber die Zeichen für Multiplikation und Division sind vielleicht neu für Sie.

+ Addition
− Subtraktion
* Multiplikation
/ Division
= Gleichheitszeichen

Die Taste für das Einführungszeichen = Potenzierungszeichen (Λ) wird auch für mathematische Operationen verwendet.

[SHIFT Λ]

Diese Taste gibt dem Computer an, daß eine Potenzierung (eine Zahl wird zur Potenz erhoben) durchzuführen ist. Da 5^3 auf dem Bildschirm schwer zu drucken ist, interpretiert der Computer in der Folge 5 Λ 3 die Zahl 3 als Exponenten.

Die folgenden Tasten werden zur Angabe der mathematischen Beziehungen in TI-BASIC verwendet:

[SHIFT >] "größer als"; dieses Symbol dient zum Vergleich zweier Größen.
[SHIFT <] "kleiner als"; mit diesem Symbol werden ebenfalls 2 Größen verglichen.

Leertaste

Die Leertaste ist die lange, unbeschriftete Schiene, die den unteren Teil der Tastatur bildet. Sie funktioniert wie die Leertaste einer Standard-Schreibmaschine. Wenn Sie die Leertaste drücken, läßt der Computer einen Zwischenraum zwischen Wörtern, Buchstaben oder Zahlen.

Mit der Leertaste können auch Zeichen, die bereits auf dem Bildschirm sind, wieder gelöscht werden (siehe hierzu den Abschnitt "Fehlerkorrektur").

Fehlerkorrektur

Sie können vor Drücken der Taste **[ENTER]** einen Tippfehler mühelos korrigieren. Führen Sie den Positionszeiger an das Zeichen zurück, das Sie ändern wollen **[FCTN ←]**. Dann tippen Sie das (die) korrekte(n) Zeichen und bewegen den Positionszeiger mit der Vorwärtstaste wieder an das Ende des Wortes oder Satzes, wo Sie stehengeblieben waren.

Sie können Fehler auch mit der Leertaste beseitigen. Gehen Sie mit **[FCTN ←]** an den Punkt zurück, wo Sie mit der Löschung beginnen wollen. Dann drücken Sie die Leertaste, um den Positionszeiger über die Zeichen auf dem Bildschirm zu führen.

Mit dieser Maßnahme werden die Zeichen gelöscht. In bestimmten Fällen können Sie auch Korrekturen mit den Tasten **[DELete]** und **[INSert]** vornehmen.

Systemerweiterung

Systemerweiterung

Für die nutzbringende Anwendung Ihres Computers ist ein breites Angebot an Erweiterungsbausteinen lieferbar.

Mit diesen Peripheriegeräten können Sie Ihre Computer-Grundausstattung jederzeit Ihren Anforderungen an ein leistungsfähiges Computer-System anpassen und somit die Möglichkeiten des TI-Home-Computers enorm erweitern.

Ob Sie die Speicherkapazität vergrößern, die Ergebnisse ausdrucken oder den Computer zum Sprechen bringen wollen, mit dem umfangreichen Peripherie-Programm zum TI 99/4A ist das kein Problem. – Er wächst mit Ihren Anforderungen mit.

TI Fernbedienung

Die leichte und handliche zweifache Fernbedienung steigert Ihre Unabhängigkeit und Vielseitigkeit bei Spielen oder grafischen und akustischen Funktionen Ihres Computers, ohne Bedienung der Tastatur. Die Fernbedienung kann in Verbindung mit bestimmten Software-Anwendungen oder eigenen BASIC-Programmen verwendet werden.

TI Sprachsynthesizer

Der Sprachsynthesizer verleiht Ihrem TI Computer eine eigene Stimme und macht die Beschäftigung mit dem Computer durch gesprochene Wörter, Ausdrücke und Sätze noch amüsanter und lehrreicher (Fremdsprachen-Lernen). Um den Sprachsynthesizer zu aktivieren, benötigt man geeignete Programm-Module. Sie können zum Beispiel das Sprachmodul, das Programm-Modul "Terminal-Emulator II", oder ein beliebiges anderes auf Sprache programmiertes Modul verwenden.

Disketten-System

Das Disketten-System ist ein Großspeichersystem, bestehend aus der Diskettensteuerung und einem bis drei Diskettenantrieben. Mit diesem System können Sie Ihre Computerprogramme zur späteren Wiederverwendung speichern oder die auf Diskette lieferbaren Programme nutzen. Daneben sind einige der Programm-Module so angelegt, daß Sie Daten und Rechenergebnisse ebenfalls auf Diskette speichern können.

Der Programm-Modul "Disk-Manager" ist im Lieferumfang jeder Diskettensteuerung enthalten. Mit diesem Modul kann man eine Diskette katalogisieren, initialisieren, Disketten oder Dateien benennen, Dateien löschen, Disketten oder Dateien kopieren, Dateien schützen, und die einwandfreie Funktion des Speichersystems testen.

TI Drucker

Wenn Sie den Drucker an Ihren Computer anschließen, können Sie eine gedruckte Kopie Ihrer Programme und Daten anfertigen lassen, zum Beispiel als Hilfe für die Überprüfung langer Programme oder um Dateien von Programmen und Resultaten aufzubewahren.

Daneben kann man den Drucker in Verbindung mit einigen Software-Anwendungen einsetzen, um Bildschirmanzeigen auszudrucken, oder um gedruckte Listen und Protokolle zu erhalten.

Das Gerät druckt bis zu 80 Zeichen pro Zeile.

TI V. 24 Schnittstelle

Die V. 24 Schnittstelle von Texas Instruments ermöglicht den Anschluß einer Vielzahl von V. 24-kompatiblen Zubehörgeräten an Ihren Computer. Bei angeschlossener V. 24 Schnittstelle können Sie Programme mit einem breiten Drucker mit z. B. 80 Zeichen pro Zeile auflisten, ein Modem zur Datenfernübertragung ansprechen, grafische Darstellungen auf einem Plotter erzeugen und vieles mehr.

Akustik-Koppler (Modem)

Mit einem Akustik-Koppler, der an die V. 24 Schnittstelle angeschlossen ist, kann der Computer über normale Telefonleitungen mit einer anderen ähnlichen Einrichtung kommunizieren.

Wenn Sie auch einen Programm-Modul für Datenfernübertragung haben, ist der Zugriff auf Datenbankdienste möglich.

TI Speichererweiterung

Mit der Speichererweiterung wird der Arbeitsspeicher Ihres Computers um 32 K Byte RAM vergrößert. Außerdem wird damit die Anzahl der Zubehörgeräte erhöht, die an den Computer angeschlossen werden können. (Anmerkung: Die Speichererweiterung erfordert eine Programm-Modul z. B. "Extended-Basic" oder ein Peripheriegerät, das speziell auf diesen Speicherbereich zugreift. Die in den Computer eingebaute TI BASIC Programmiersprache kann von der Speichererweiterung keinen Gebrauch machen.)

Anschluß eines Kassettenrekorders

Kassettenrekorder-Anschlußkabel*

Eine zusätzliche Erweiterung des Computersystems ist durch den Einsatz von Kassettenrekordern möglich. Mit TI-BASIC können Sie Daten, die Sie in den Computer eingeben (Programme, numerische Daten etc.) speichern und wiedereinlesen. Wenn Sie eine Bandaufnahme der Daten machen, können diese als permanente Aufzeichnung aufbewahrt werden. Später können Sie dann die Daten von der Kassette in den Speicher des Computers laden, wenn Sie die Informationen wieder verwenden möchten. Bei Einsatz von mehreren Moduln kommt diese Eigenschaft, Daten zu speichern und wieder zu laden, zum Tragen.

Sie können einen oder zwei Kassettenrekorder einsetzen. Die Verwendung von zwei Kassettenrekordern ist vor allem bei komplizierten Programmen von Vorteil.

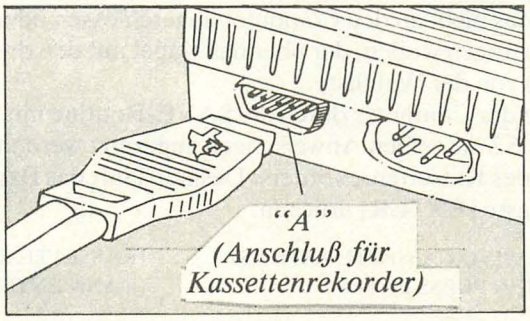
Der Anschluß der Kassettenrekorder an den Home-Computer erfolgt mit dem Kassettenrekorder-Anschlußkabel (Zubehör). Viele Standard-Kassettenrecorder sind für den Home-Computer geeignet. Im Interesse einer optimalen Funktion sollten jedoch folgende Voraussetzungen gegeben sein:

- Lautstärkeregler
- Klangregler
- Mikrofonbuchse
- Fernbedienungsbuchse
- Anschluß für Kopfhörer oder externe Lautsprecher
- digitales Bandzählwerk (damit finden Sie leicht die korrekte Bandstelle, wenn Sie mehr als ein Programm oder mehrere Dateien auf demselben Band speichern wollen).

Anmerkung: Beim Anschlußkabel wird der 3polige Stecker für Rekorder 1 "CS1" verwendet, der 2polige für Rekorder "CS2". CS1 kann zum Speichern und Wiedereinlesen verwendet werden, CS2 nur zum Speichern.

Um Ihre(n) Kassettenrekorder an den Computer anzuschließen, verwenden Sie nur das oben erwähnte Anschlußkabel und halten sich an die folgenden einfachen Schritte.

1. Stecken Sie den 9poligen Stecker des Kabels in den dafür vorgesehenen 9poligen Anschluß der Konsole "A".



2. Die drei freien Klinkenstecker des Kabels werden dann wie folgt mit dem (den) Kassettenrekorder(n) verbunden:
   - der Stecker mit dem roten Kabel wird in die Mikrofonbuchse gesteckt;
   - der Stecker mit dem schwarzen Kabel ist für die Fernbedienungsbuchse vorgesehen (beachten Sie, daß dieser Stecker kleiner ist als die beiden anderen);
   - der Stecker mit dem weißen Kabel wird mit dem Anschluß für Kopfhörer (oder externe Lautsprecher) verbunden.
3. Wir bezeichnen den ersten Kassettenrekorder als "CS1" und den zweiten als "CS2", um Verwechslungen zu vermeiden, welches Gerät gerade benutzt werden soll. Die Klinkenstecker sind mit 1 und 2 beschriftet, um Gerät CS1 oder CS2 anzugeben. Achten Sie bei der Aufnahme (Abspeichern auf Kassette) oder beim Laden der Daten (Wiedereinlesen der Daten in den Computer) darauf, wie die Kassettenrekorder angeschlossen sind. Zum Laden eignet sich nur CS1. Weitere Informationen im entsprechenden Abschnitt.

Anmerkung: Wenn Sie nur einen Kassettenrekorder anschließen, bleibt das freie Kabelende unbenutzt.

Nach Anschluß aller Kabel stellen Sie den Klangregler auf Ihrem Kassettenrekorder zwischen halbe und volle "Höhen". Für die Lautstärke wählen Sie eine mittlere Stellung (wenn der Lautstärkeregler eine Zehnerskala hat, schieben Sie ihn auf 5 oder auf die in der Tabelle genannte Position).

Hat Ihr Kassettenrekorder keinen Klangregler, müssen Sie unter Umständen die Lautstärke höher einstellen, um bessere Ergebnisse zu erzielen.

* separat als Zubehör

Kassettenrecorder als externer Speicher

Übertragung / Eingabe von Daten

Wenn Sie Ihre(n) Kassettenrekorder anweisungsgemäß an die Konsole angeschlossen haben, können Sie Daten übertragen/eingeben.

Ehe Sie versuchen, Ihre Daten auf Kassette zu übertragen oder in den Computer einzugeben, achten Sie auf folgende Punkte:

- Benutzen Sie ausschließlich Tonbänder von hoher Qualität. Schlechte Qualität führt zu schlechten Resultaten.
- Verwenden Sie max. C-60-Bänder. Längere Bänder sind dünner und haben geringere Klangqualität.
- Der Kassettenrekorder muß mindestens 65 cm Abstand vom Bildschirm oder vom Fernsehgerät haben, um magnetische Störfelder zu vermeiden.
- Das Band muß mindestens ca. 65 cm Abstand vom Bildschirm, vom Fernsehgerät oder von einem Elektromotor bzw. von anderen starken Magnetfeldern haben, um eine zufällige Löschung Ihrer Daten zu vermeiden.
- Das gesamte System (Computerkonsole, Kassettenrekorder und der Farbmonitor) darf nicht auf einer durchgehenden Metalloberfläche stehen, um das Leitungsrauschen minimal zu halten.
- Wenn Sie zwei Kassettenrekorder verwenden, müssen Marke und Modell identisch sein. Die Anwendung von verschiedenen Kassettengeräten kann wegen der Unterschiede in der Massebeschaltung eines oder beide Geräte auf Dauer beschädigen. Verwenden Sie zum Laden nur CS1. Zum Speichern können CS1 wie CS2 benutzt werden.

Übertragung / Eingabe von Daten in TI-BASIC

Vollständige Instruktionen zur Eingabe und Übertragung von Daten, wenn Sie TI-BASIC programmieren, finden Sie auf den Seiten 51 bis 53 dieser Anleitung.

Datenübertragung bei Anwendung eines Moduls

Nach Eingabe Ihrer Daten in den Computer und nach Anschluß des Kassettenrekorders (mit eingelegter Kassette von hoher Qualität) können Sie mit der Aufzeichnung beginnen.

Wählen Sie die "SAVE"-Funktion, die das von Ihnen verwendete Modul bietet.

Der Computer bietet dann eine Auswahlliste für die Datenübertragung an. (Anmerkung: Sie erhalten eine Fehlermeldung, wenn Sie ein Gerät wählen, das nicht an den Computer angeschlossen oder eingeschaltet ist.) Wenn Sie Ihre Daten auf den Kassettenrekorder übertragen wollen, der über das Kabel mit den drei Klinkensteckern angeschlossen ist, wählen Sie CS1 (Kassettenrekorder 1) von der Wahlliste.

Von diesem Augenblick an führt Sie der Computer durch die SAVE-Routine mit Anweisungen auf dem Bildschirm. (Bitte beachten Sie, daß für CS1 und CS2 die selben Anweisungen angezeigt werden.)

Der Computer steuert den Antrieb des Kassettenrekorders. Deshalb läuft das Band nicht an, bevor Sie entsprechend der Anzeige auf dem Bildschirm die Taste **[ENTER]** drücken.

Bildschirm-Instruktionen	REWIND CASSETTE TAPE CS1	(KASSETTE RUECKLAUF CS1)
	THEN PRESS ENTER	(DANN ENTER DRUECKEN)

Verfahren

Spulen Sie das Band zurück, ehe Sie **[ENTER]** drücken. Wenn Ihr Kassettenrekorder kein Bandzählwerk hat, spulen Sie das Band an den Anfang zurück.

Ist ein Bandzählwerk vorhanden, spulen Sie das Band an die Stelle, wo Sie mit der Aufzeichnung beginnen wollen, und drücken Sie die "Stop"-Taste auf dem Rekorder.

(Notieren Sie sich die Position.) Dann drücken Sie **[ENTER]**, um fortzufahren.

PRESS CASSETTE RECORD CS1	(KASSETTE AUFNAHME CS1)
THEN PRESS ENTER	(DANN ENTER DRUECKEN)

Drücken Sie den Aufnahmeknopf auf dem Rekorder, und dann **[ENTER]** auf dem Computer. Sobald dies geschehen ist, beginnt die Aufzeichnung Ihrer Daten auf Band, und der Bildschirm zeigt folgende Nachricht an:

RECORDING	(AUFNAHME)

Während der Speicherung auf Band oder dem Einlesen der Daten in den Computer können Sie den Ton der kodierten Informationen hören. Zusätzlich werden mehrere Sekunden Leerband aufgezeichnet, um ein Vorspannband mit einzukalkulieren.

PRESS CASSETTE STOP CS1	(KASSETTE STOP CS1)
THEN PRESS ENTER	(DANN ENTER DRUECKEN)

Wenn alle Daten aufgezeichnet sind, drücken Sie die Stop-Taste auf dem Rekorder und dann die Taste **[ENTER]** auf dem Computer.

Daraufhin erscheint die Frage: CHECK TAPE (Y OR N)? (BANDKONTROLLE (J ODER N)?

Kassettenrecorder als externer Speicher

Anmerkung:

Die Antworten, die nur aus einem Buchstaben bestehen (Y, N, R etc.), welche Sie bei der Übertragung oder Eingabe von Daten eintippen, müssen großgeschrieben werden. Drücken Sie die **[SHIFT]**-Taste und gleichzeitig die erforderliche Buchstabentaste.

An dieser Stelle haben Sie die Möglichkeit, das Band durch den Computer kontrollieren zu lassen, um sicherzustellen, daß alles ordnungsgemäß aufgezeichnet wurde. Wir empfehlen dringend, diese Möglichkeit zu nutzen, um die Genauigkeit des Bandes für den späteren Gebrauch zu gewährleisten. Beachten Sie: **nur CS1** verwenden.

Wenn Sie das Band nicht kontrollieren wollen, drücken Sie N für Nein. Nehmen Sie das Band aus dem Rekorder und beschriften Sie es für die spätere Verwendung.

Falls Sie sich für die Bandkontrolle entscheiden, drücken Sie Y für YES (JA). Sie werden wiederum vom Computer mit folgenden Meldungen angeleitet:

Bildschirm-Instruktionen	REWIND CASSETTE TAPE CS1	(KASSETTE RUECKLAUF CS1)
	THEN PRESS ENTER	(DANN ENTER DRUECKEN)

Verfahren

Spulen Sie das Band vor Drücken der Taste **[ENTER]** an die Stelle zurück, wo die Aufzeichnung Ihrer Daten beginnt. Wenn Sie die Daten vom Bandanfang an aufgenommen haben, spulen Sie das Band zum Anfang zurück. Beginnt die Aufzeichnung jedoch an anderer Stelle, spulen Sie das Band bis zu dieser Position zurück, drücken die Stop-Taste auf dem Rekorder und dann die Taste **[ENTER]**.

PRESS CASSETTE PLAY CS1	(KASSETTE WIEDERGABE CS1)
THEN PRESS ENTER	(DANN ENTER DRUECKEN)

Drücken Sie den Wiedergabeknopf auf dem Rekorder und dann die Taste **[ENTER]**. Der Computer vergleicht nun die Daten in seinem Speicher mit den Daten auf dem Band. Während Ihr Band vom Computer überprüft wird, sehen Sie am Bildschirm die Nachricht:

CHECKING	(KONTROLLE)

Liegen keine Fehler vor, werden folgende Meldungen am Bildschirm angezeigt:

*DATA OK	(*DATEN OK)
PRESS CASSETTE STOP CS1)	(KASSETTE STOP CS1)
THEN PRESS ENTER	(DANN ENTER DRUECKEN)

Sie können nun Ihre Kassette herausnehmen und sie für die spätere Verwendung beschriften. Wurden die Daten jedoch nicht richtig aufgezeichnet, erhalten Sie eine der beiden folgenden Fehlermeldungen:

Fehlermeldung	*ERROR – NO DATA FOUND	(*FEHLER – KEINE DATEN

Bedeutung

Ihre Daten wurden nicht aufgezeichnet oder nicht abgespielt (wiedergegeben).

PRESS R TO RECORD CS1	(R FUER CS1 AUFNAHME DRUECKEN)
PRESS C TO CHECK	(C ZUR KONTROLLE DRUECKEN)
PRESS E TO EXIT	(E FUER "BEENDIGUNG DES VORGANGES" DRUECKEN)

oder

*ERROR IN DATA DETECTED	(*FEHLER IN DEN DATEN)

Bedeutung

Ein Teil Ihrer Daten wurde nicht richtig aufgezeichnet.

PRESS R TO RECORD CS1	(R FUER CS1 AUFNAHME DRUECKEN)
PRESS C TO CHECK	(C ZUR KONTROLLE DRUECKEN)
PRESS E TO EXIT	(E FUER "BEENDIGUNG DES VORGANGES" DRUECKEN)

Ehe Sie fortfahren, sollten Sie folgende Punkte prüfen:

• Hat der Rekorder einen ausreichenden Abstand vom Fernsehgerät (mindestens 65 cm)? • Ist er richtig angeschlossen? • Ist das Band der Kassette in gutem Zustand? (Versuchen Sie im Zweifelsfall ein anderes.) • Ist die Lautstärke zu hoch oder zu gering? • Muß der Magnetkopf des Rekorders gereinigt werden? • Steht das System auf einer metallischen Oberfläche?

Kassettenrecorder als externer Speicher

Wenn Sie Ihre Kontrolle im Hinblick auf mögliche Störungsquellen abgeschlossen haben, können Sie mit einer der oben genannten Maßnahmen fortfahren:

- Drücken Sie **[R]**, um Ihre Daten nach den obigen Instruktionen für die Aufnahme erneut aufzuzeichnen.
- Drücken Sie **[C]**, um den Computer zu einer erneuten Kontrolle der Daten anzuweisen.
- Drücken Sie **[E]**, um an den Ausgangspunkt zurückzukehren. Die folgende Nachricht erscheint auf dem Bildschirm:

*PRESS CASSETTE STOP CS1	(KASSETTE STOP CS1)
THEN PRESS ENTER	(DANN ENTER DRUECKEN)

Mit der "E"-Taste gehen Sie an den Anfang der "SAVE"-Funktion des Moduls zurück. Wenn Sie dann **[ENTER]** drücken, können Sie die Speicherung Ihrer Daten erneut versuchen. Gehen Sie einfach nach den Instruktionen auf dem Bildschirm vor.

Dateneingabe bei Anwendung eines Moduls

Wenn Sie die auf Band gespeicherten Informationen später verwenden wollen, müssen Sie Ihre Daten "laden", d. h. die Daten, die Sie auf Band übertragen haben, müssen in den Arbeitsspeicher des Computers eingelesen werden*. Zunächst schließen Sie Ihre(n) Kassettenrecorder an den Computer an. Dann schieben Sie das Modul ein, aus dem Sie die Daten aufgezeichnet haben. Wenn Sie zum Einlesen der Daten bereit sind, wählen Sie die "LOAD-DATA"-Funktion des Moduls (Daten einlesen). Bei der Dialogeröffnung des Computers drücken Sie **[1]**, um anzugeben, daß die Daten von einer Kassette eingelesen werden sollen. Dann drücken Sie für die Wahl von Kassettengerät 1 (CS1) erneut **[1]**. Erinnern Sie sich: zum Laden wird CS1 benutzt.

Bildschirm-Instruktionen

REWIND CASSETTE TAPE CS1	(KASSETTE RUECKLAUF CS1)
THEN PRESS ENTER	(DANN ENTER DRUECKEN)

Spulen Sie das Band vor Drücken der Taste **[ENTER]** zurück, und zwar bis an die Stelle, von der aus Sie Daten in den Computer einlesen wollen (wenn der Kassettenrekorder kein Bandzählwerk hat, wird an den Beginn zurückgespult). Dann drücken Sie **[ENTER]**.

* *Wegen der Unterschiede im Design der Kassettenrekorder kann ein Band, das von einem Modell aufgenommen wurde, nicht mit einem anderen Modell eingelesen werden.*

Bildschirm

PRESS CASSETTE PLAYCS1	(KASSETTE WIEDERGABE CS1)
THEN PRESS ENTER	(DANN ENTER DRUECKEN)

Verfahren

Drücken Sie den Wiedergabeknopf auf dem Rekorder und dann die Taste **[ENTER]**. Die Informationen werden vom Band gelesen und in den Speicher des Computers eingegeben. Während der Computer das Band einliest, erscheint folgende Meldung auf dem Schirm:

READING	(EINLESEN)

Das Einlesen der Daten erfordert etwas Zeit, je nach der Menge der gespeicherten Informationen. Wenn der Einlesevorgang abgeschlossen ist, gibt der Computer Auskunft, ob die Daten korrekt eingelesen wurden. In diesem Fall sehen Sie folgendes auf dem Bildschirm:

DATA OK	(DATEN OK)
PRESS CASSETTE STOP CS 1	(KASSETTE STOP CS 1)
PRESS CASSETTE ENTER	(DANN ENTER DRUECKEN)

Sie können nun die Arbeit mit dem Modul aufnehmen. Wenn jedoch die Daten nicht richtig in den Speicher des Computers eingegeben wurden, erscheint eine Fehlermeldungen auf dem Bildschirm. Gehen Sie nach den Anweisungen auf dem Schirm vor, um die Eingabe Ihrer Daten erneut zu versuchen. Bleiben die Schwierigkeiten bestehen, stellen Sie folgende Punkte sicher:

- Sie lesen das richtige Band ein.
- Das Band ist auf den korrekten Startpunkt für die Eingabedaten gespult.
- Das Band ist nicht beschädigt oder zufällig gelöscht.
- Der Rekorder hat den richtigen Abstand vom Fernsehgerät (mindestens 65 cm).
- Der Rekorder ist sachgemäß an den Computer angeschlossen.
- Die Lautstärke des Rekorders ist korrekt eingestellt.
- Das System steht nicht auf einer metallenen Oberfläche.
- Das Band wurde mit Ihrem eigenen Rekorder oder mit dem gleichen Modell aufgenommen.
- Der Magnetkopf des Kassettenrekorders ist sauber.
- Sie arbeiten mit Kassettenrekorder Nummer 1.

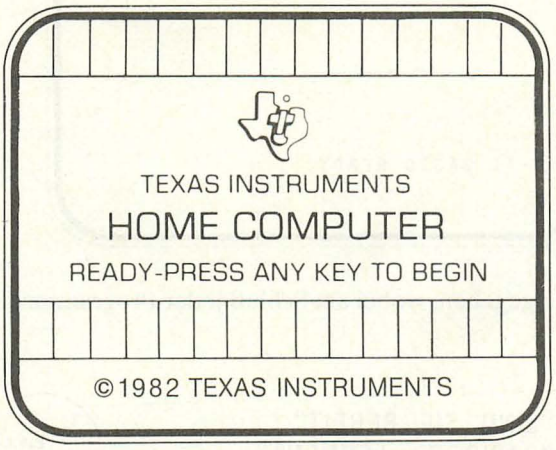
Einfaches Programmieren

Dieses Kapitel zeigt Ihnen – wenn Sie bisher noch nie mit einem Computer gearbeitet haben – mit leicht nachvollziehbaren Beispielen, wie einfach es ist, mit dem TI-Home-Computer zu "reden".

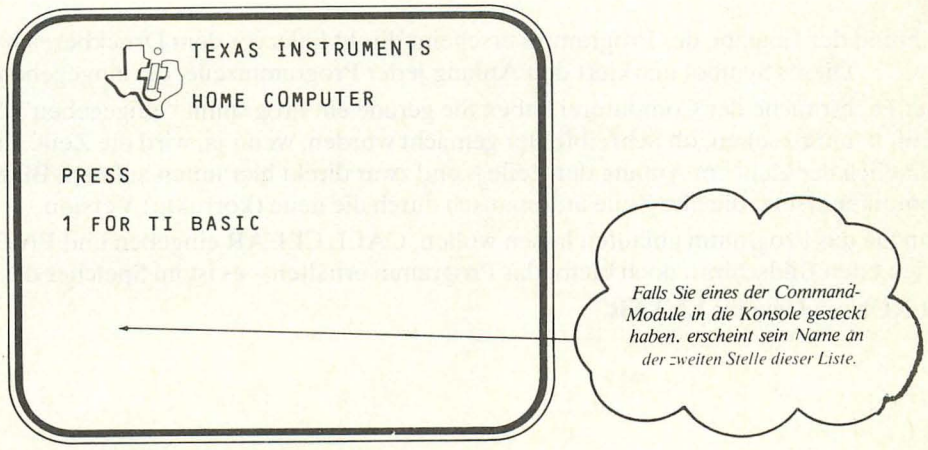
Um sich überhaupt mit einem Computer in Verbindung setzen zu können, müssen Sie seine Sprache erlernen – die einfache, aber sehr leistungsfähige Computersprache TI-BASIC.

Damit Sie aber erst einmal mit dem Computer vertraut werden, wollen wir lernen, wie man einen Computer programmiert.

Im Direktbetrieb führt Ihr Computer jede von Ihnen über die Tastatur eingegebene Basic-Anweisung "direkt" durch, sobald Sie **ENTER** (Eingabe) drücken. Da Sie auf dem Bildschirm eine sofortige Reaktion sehen, ist der Direktbetrieb ein praktisches Mittel, bestimmte Anweisungen der BASIC-Programmiersprache vorzustellen und näher zu betrachten. Wenn Sie bereit sind, schalten Sie Ihren Computer ein. Daraufhin erscheint auf dem Bildschirm das Titelbild:



Drücken Sie jetzt eine beliebige Taste. Daraufhin erscheint auf dem Bildschirm die Hauptwahlliste:



(*Anmerkung:* Wenn Sie TI-BASIC wieder verlassen wollen, tippen Sie einfach das Wort **BYE** (GOOD **BYE** = auf Wiedersehen) und drücken die **ENTER**-Taste (Eingabe). Der Bildschirm zeigt dann wieder die Hauptwahlliste.)

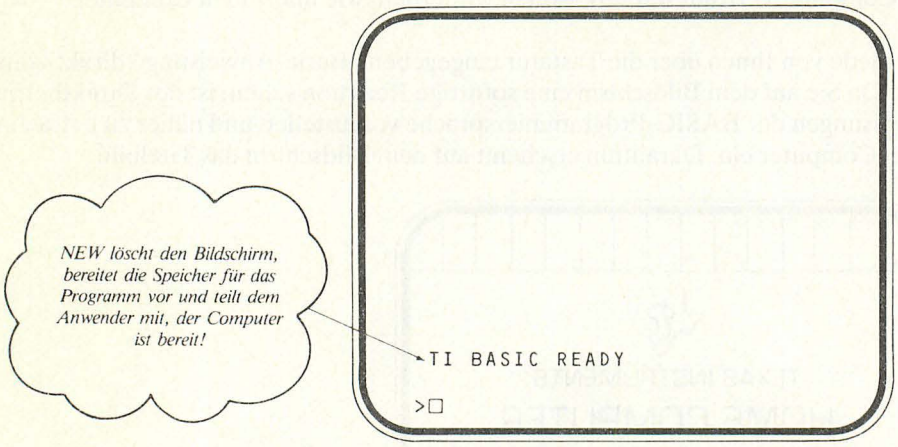
Die Beispiele in diesem Buch sind in Großschreibung ausgedruckt. Wenn Sie diese Beispiele genau nachvollziehen wollen, drücken Sie die **ALPHA LOCK**-Taste. In den meisten Fällen akzeptiert der Computer jedoch Groß- und Kleinschreibung.

Drücken Sie nun die Taste 1, um TI-BASIC zu wählen. Der Bildschirm zeigt jetzt, daß der Computer eingabebereit ist.

Einfaches Programmieren

Ein Programm für das Ausdrucken

Beginnen wir mit der PRINT-Anweisung in einem Programm. Zuerst wird das Wort NEW eingegeben und dann **ENTER** gedrückt.



Jetzt wird das folgende Programm eingegeben, wobei am Schluß jeder Programmzeile **ENTER** zu drücken ist:

```
10 PRINT "SIND SIE BEREIT"
20 PRINT "BASIC ZU LERNEN?"
30 END
```

Ein Leerzeichen

(Während der Eingabe des Programms erscheint direkt links vor dem Druckbereich ein kleines "Aufforderungszeichen" >. Dieses Symbol markiert den Anfang jeder Programmzeile, die eingegeben wird.)

In der Fachsprache der Computerei haben Sie gerade ein Programm "eingegeben". Sie sollten das Programm jetzt überprüfen, um zu sehen, ob Schreibfehler gemacht wurden, wenn ja, wird die Zeile einfach korrekt neu eingegeben: einschließlich der Zahl am Anfang der Zeile – und zwar direkt hier unten auf dem Bildschirm. Dann **ENTER** drücken: der Computer ersetzt die alte Zeile automatisch durch die neue (korrekte) Version.

Wenn Sie das Programm ablaufen lassen wollen, CALL CLEAR eingeben und **ENTER** drücken: Der Computer löscht jetzt den Bildschirm, doch bleibt das Programm erhalten – es ist im Speicher des Computers abgelegt!

Jetzt RUN und wieder **ENTER**.

Einfaches Programmieren

```
>RUN
 SIND SIE BEREIT,
 BASIC ZU LERNEN?
  ** DONE **

>□
```

Soll das Programm noch einmal "laufen"? Einfach wieder RUN eingeben und **ENTER** drücken.

```
>RUN
 SIND SIE BEREIT,
 BASIC ZU LERNEN?
  ** DONE **

>RUN
 SIND SIE BEREIT,
 BASIC ZU LERNEN?
 ** DONE **
>□
```

Jedesmal, wenn RUN über die Tastatur eingegeben und **ENTER** gedrückt wird, beginnt der Rechner mit der ersten Anweisung und folgt den Anweisungen der Reihe nach, bis er die letzte erreicht. END bedeutet genau das: Ende, Stop!

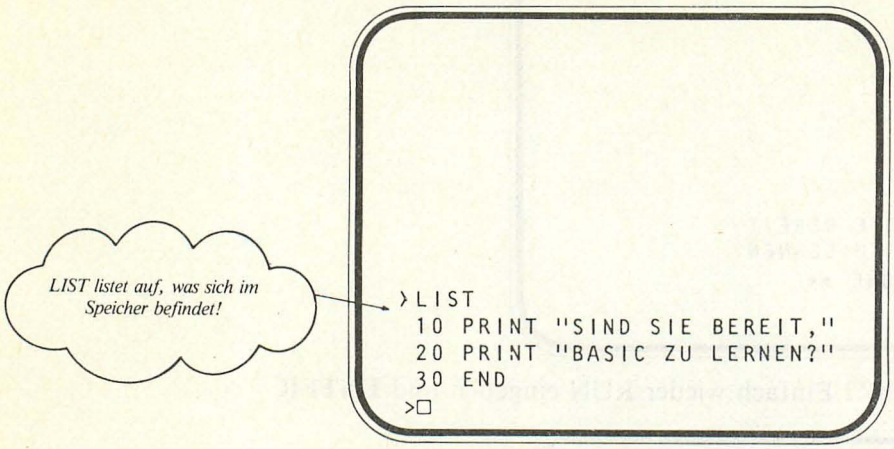
Haben Sie bemerkt, daß der Bildschirm kurz grün aufleuchtete, während das Programm lief? Während der Ausführung eines Programmes leuchtet der Schirm stets grün auf und erscheint danach in seiner normalen blauen Farbe, wenn das Programm abgeschlossen ist.

Programmaufbau

Nach diesen ersten Programmiererfahrungen werden wir jetzt einige der Dinge näher betrachten, die bei der Eingabe des obigen Programmes zugrunde lagen. Des besseren Verständnisses wegen sei zunächst das Programm zurück auf den Bildschirm geholt:

Einfaches Programmieren

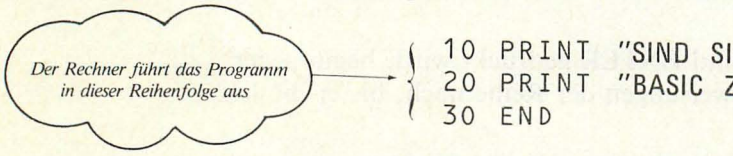
Zuerst wird CALL CLEAR (ohne eine Zeilennummer) eingegeben und dann **ENTER** gedrückt, um den Bildschirm zu löschen. Jetzt tippt man LIST und drückt dann wiederum **ENTER**:



Das obige Programm besteht aus drei Anweisungen oder "Zeilen". Jede Anweisung beginnt mit einer Zeilennummer, die zwei wichtige Funktionen hat.

1. Sie teilt dem Computer mit, die Anweisung nicht direkt auszuführen, sondern sie nach Drücken der **ENTER**-Taste im Speicher abzulegen.
2. Sie legt die Reihenfolge fest, in der die Anweisungen im Programm ausgeführt werden.

Wie im Direkbetrieb wird am Schluß der Eingabe jeder Programmzeile **ENTER** gedrückt. Durch das Drücken der **ENTER**-Taste wird das Ende der Programmzeile definiert, genauso wie die Zeilennummer den Anfang der Zeile kennzeichnet. Für den Rechner ist dies auch ein Hinweis, die Zeile abzuspeichern. Am Ende jeder Programmzeile muß **ENTER** gedrückt werden - geschieht dies nicht, speichert der Rechner die Zeile nicht.



Vielleicht wundern Sie sich, warum wir die Zeilen in Zehnerschritten (10,20,30 etc.) durchnumeriert haben. Man hätte sie genauso leicht auch mit 1,2,3 numerieren können. Doch erlauben die Zehnerschritte - oder andere Schritte wie 100,200,300 etc. - neue Zeilen einzuschieben, falls das vorliegende Programm erweitert werden soll, und es ist dann nicht erforderlich, das gesamte Programm neu einzugeben (Wir kommen auf diese Technik noch einmal zurück, wenn das Editieren eines Programmes besprochen wird.)

Befehle - NEW, RUN, LIST

Diese Befehle haben wir bereits verwendet, doch ist es vielleicht angebracht, an dieser Stelle insbesondere die drei obigen noch etwas näher zu definieren.

Befehle (Commands) unterscheiden sich von Anweisungen (Statements). Sie sind nicht Teil des Programms und haben keine Zeilennummern. Ihre Aufgabe ist es, die Ausführung bestimmter Aufgaben sofort zu veranlassen.

NEW - Sagt dem Rechner, das Programm in seinem Speicher zu löschen. (Dieser Befehl löscht auch den Bildschirm, nicht mit CALL CLEAR zu verwechseln, da letztere Anweisung nur den Bildschirm löscht, nicht aber den Speicher).

Einfaches Programmieren

RUN – Sagt dem Computer, das Programm in seinem Speicher auszuführen ("ablaufen") zu lassen.
LIST – Sagt dem Computer, das Programm in seinem Speicher auf dem Bildschirm zu zeigen (oder "aufzulisten").

NEW wird also nur dann verwendet, wenn der Computer für das Abspeichern eines neuen Programmes vorbereitet werden soll. Vorsicht bei der Anwendung von NEW; im Zweifelsfalle zuerst LIST verwenden, damit das derzeitige Programm gelesen werden kann, bevor es gelöscht wird.

LIST ist eine vielseitige Hilfe für die Korrektur oder Änderung eines Programmes. Mit diesem Befehl läßt sich das Programm direkt auf den Bildschirm bringen, wo es überprüft werden kann, beziehungsweise sich etwaige Fehler beseitigen lassen.

RUN führt das Programm aus.

Ein Computer-Programm

Ihr Home Computer kann nicht nur ausdrucken oder "Nachrichten" wiedergeben, sondern hat auch beträchtliche rechnerische Fähigkeiten. Versuchen wir ein mathematisches Programm zu erstellen. Hier die Tasten, mit denen die vier grundlegenden Rechenoperationen ausgeführt werden:

+ für die Addition
– für die Subtraktion
* für die Multiplikation
/ für die Division

*Beachten Sie, daß +, – und * die Symbole für die Großschreibung sind, die über =, / und 8 stehen.*

Als Beispiel soll ein einfaches Programm geschrieben werden, das die Einheit Kilogramm in englische Pfund umwandelt (1 Kilogramm = 2,2 Pfund). Als erstes werden durch Eingabe von NEW und drücken von **ENTER** der Bildschirm und die Speicher des Computers gelöscht. Die Variablen K (für Kilogramm) und P (für Pfund) sollen uns daran erinnern, um welchen Begriff es sich handelt. Wir beginnen unser Programm mit der Zuordnung der Werte zu diesen Variablen.

Eingabe:

```
10 LET K=50
20 LET P=2.2*K
```

ENTER

In diesem Fall versuchen wir herauszufinden, wieviel Pfund gleich 50 Kilogramm sind; daher wurde K als 50 definiert. Beachten Sie, daß P gleich 2,2 × K gesetzt wurde. Falls das Programm an dieser Stelle abgeschlossen und dann ausgeführt werden würde, könnte der Computer zwar die Umwandlung durchführen, das Ergebnis allerdings nicht auf dem Bildschirm darstellen! Daher brauchen wir noch die Zeile

```
30 PRINT P
```

und **ENTER**. Hat der Computer jetzt alles, was er für die Ausführung des Programms braucht? Er kennt die Anzahl der Kilogramm, die in Pfund umgerechnet werden sollen, er weiß, wie diese Umwandlung geschehen soll, und er hat auch die erforderliche Anweisung, das Ergebnis auf dem Bildschirm dazustellen. Jawohl, er hat alle erforderlichen Informationen erhalten und somit wird die Zeile

```
40 END
```

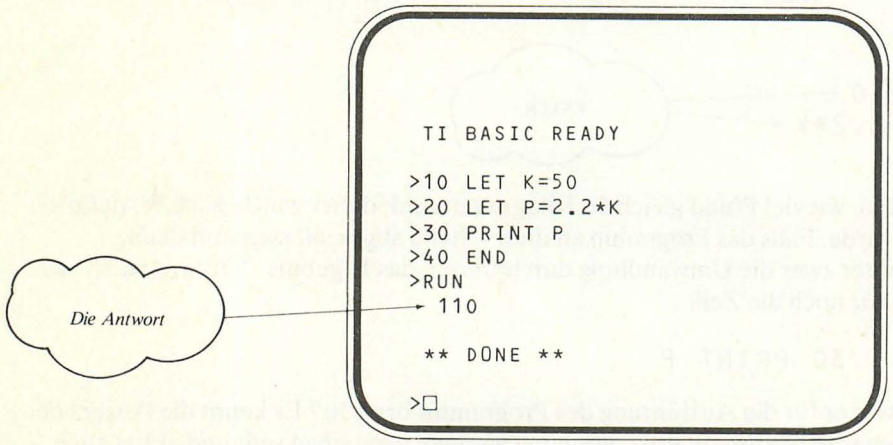
eingegeben und **ENTER** gedrückt. Das Programm sollte folgendermaßen aussehen:

Einfaches Programmieren

```
TI BASIC READY

>10 LET K=50
>20 LET P=2.2*K
>30 PRINT P
>40 END
>□
```

Bevor das Programm ausgeführt wird, seien an dieser Stelle noch zwei Eigenschaften von TI-BASIC erwähnt, durch die sich dieser "Dialekt" von anderen Versionen der Sprache unterscheidet. Erstens wird der Anfang jeder Programmzeile durch ein Aufforderungszeichen (links vom Druckfeld auf dem Bildschirm) markiert. Seine Funktion wird besser verständlich, wenn Programmzeilen eingegeben werden, die über eine einzelne Bildschirmzeile hinausreichen. Zweitens ist die END-Anweisung in einem TI-BASIC-Programm wahlweise anzuwenden. Da es sich bei dieser Anweisung jedoch um einen üblichen Teil von Basic handelt, wird sie in diesem Beispiel beibehalten.

Nun wird das Programm auf Schreibfehler überprüft. Falls erforderlich, wird die jeweilige Zeile erneut korrekt eingegeben - einschließlich der Zeilennummer und **ENTER.** Dann gibt man RUN ein und drückt **ENTER.**

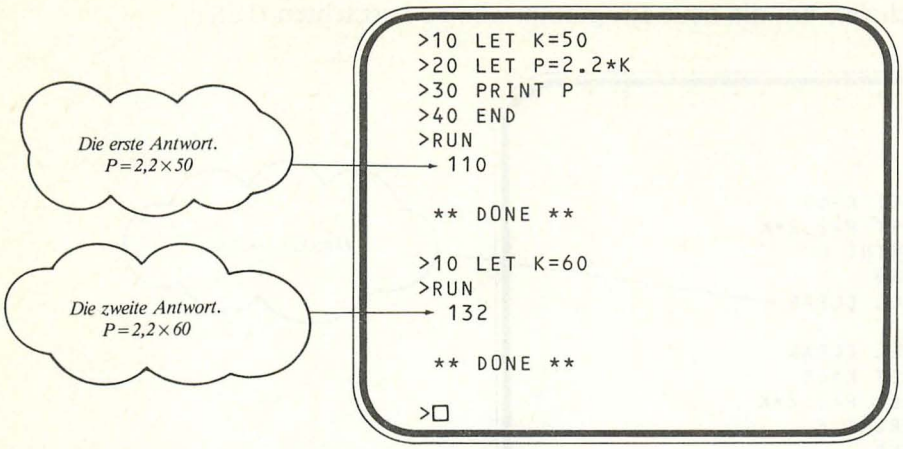


Ihre Antwort steht auf dem Bildschirm. 50 Kilogramm sind gleich 110 Pfund. Angenommen, es soll die Anzahl der Pfunde errechnet werden, die gleich 60 Kilogramm sind. Nichts leichter als das! Hierzu muß nur eine Zeile geändert werden - Zeile 10.

```
10 LET K=60
```

eingeben und **ENTER** drücken. Nun RUN tippen und wieder **ENTER** drücken.

Einfaches Programmieren



Das Editieren eines Programmes

Der gerade durchgeführte Schritt wird als "Editieren" eines Programmes bezeichnet. Die Möglichkeit, ein Programm zu editieren oder zu ändern, ohne das Ganze noch einmal neu eingeben zu müssen, ist etwas, das man in dem Maße schätzen lernt, in dem die Programmierfähigkeiten wachsen. Um einen Einblick in die große Flexibilität zu geben, die das Editieren dem Programmieren verleiht, soll das obige Programm durch ein paar weitere Änderungen umgebaut werden.

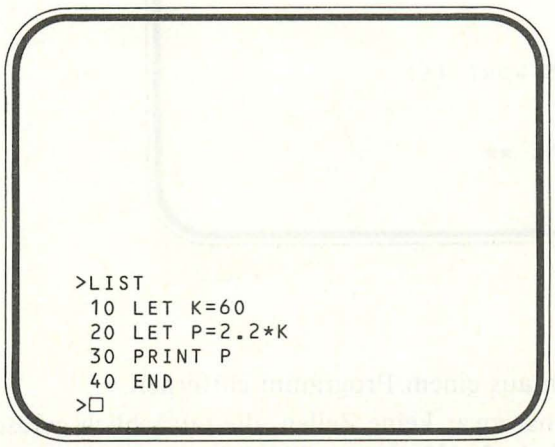
Programmzeilen hinzufügen

Wie bereits erwähnt, wurden die Programmzeilen in Zehnerschritten (anstelle von 1,2,3 etc.) durchnumeriert, damit sich Programmzeilen hinzufügen lassen, ohne daß das gesamte Programm erneut eingegeben werden muß. Bevor wir mit ein paar Beispielen experimentieren, wird der Bildschirm gelöscht und das Programm auf den Bildschirm zurückgeholt.

Eingabe:

```
CALL CLEAR
LIST
```

ENTER *drücken*



(Das Aufforderungszeichen erscheint nicht links von den Zeilen, die der Computer ausdruckt – es werden nur die Zeilen markiert, die der Anwender eingibt!)

Zunächst soll dem Programm eine CALL CLEAR-Anweisung hinzugefügt werden, damit der Bildschirm nicht jedesmal über die Tastatur gelöscht werden muß, wenn das Programm "ablaufen" soll.

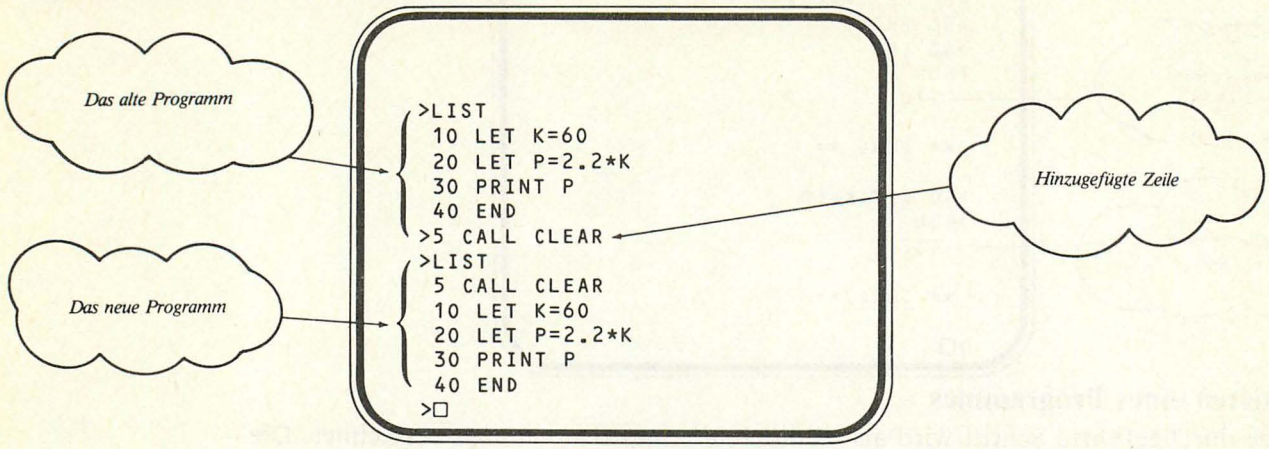
Eingabe:

```
5 CALL CLEAR
```

ENTER *drücken*

Einfaches Programmieren

Nun wird das Programm erneut aufgelistet, um die neue Programmzeile zu betrachten (LIST eingeben und **ENTER** drücken).

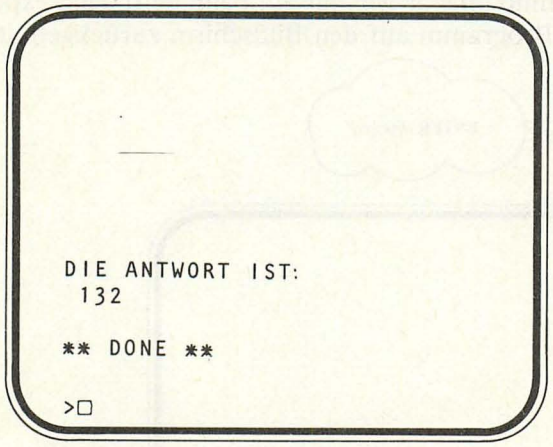


Bei einem Vergleich der beiden Programme auf dem Bildschirm fällt auf, daß der Rechner die neue Zeile automatisch an der richtigen Stelle eingeordnet hat. Lassen Sie das Programm erneut laufen, um die Wirkung der zusätzlichen Zeile zu sehen.

Nun wird eine weitere Zeile hinzugefügt, durch die die Antwort hervorgehoben wird:

```
27 PRINT "DIE ANTWORT IST:"
```

eingeben und **ENTER** drücken. Nach einer erneuten Ausführung des Programms erscheint auf dem Bildschirm:



Programmzeilen löschen

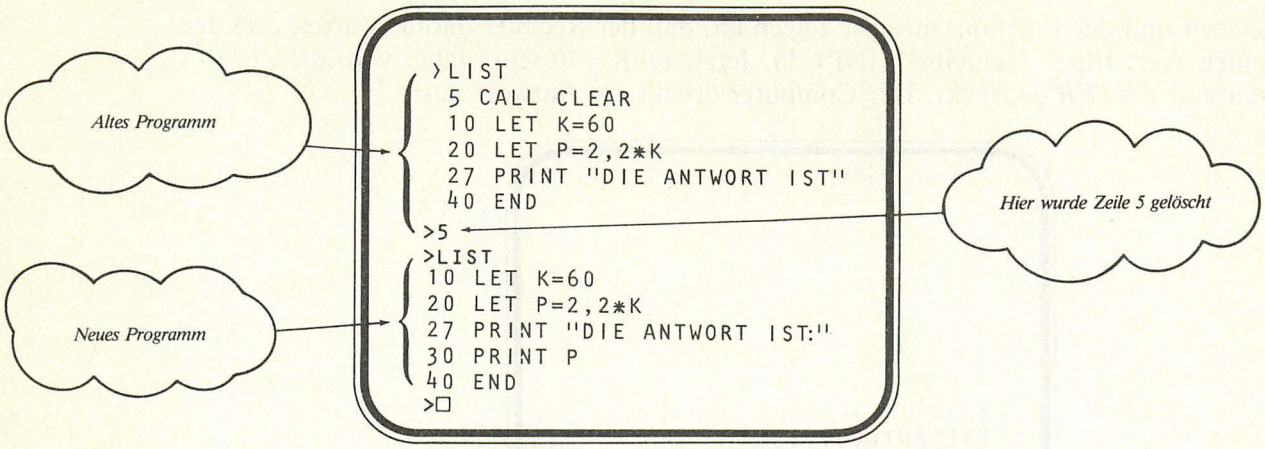
Oft muß man eine oder mehrere Zeilen aus einem Programm entfernen.

Das gerade abgespeicherte Programm hat zwar keine Zeilen, die tatsächlich gelöscht werden müßten, doch soll der Übung wegen Zeile 5 gelöscht werden.

Als erstes wird der Bildschirm gelöscht und das Programm auf dem Bildschirm ausgegeben. Zeile 5 ist die erste Zeile des Programms, eine CALL CLEAR-Anweisung. Um sie zu löschen, wird einfach Ziffer 5 eingegeben und **ENTER** gedrückt.

Danach das Programm erneut auflisten. Zeile 5 ist verschwunden!

Einfaches Programmieren



Das ist alles! Soll eine Zeile entfernt werden, muß nur die Zeilennummer eingegeben und **ENTER** gedrückt werden. Der Computer löscht dann die Zeile aus dem Programmspeicher.

Da wir Zeile 5 in diesem Programm tatsächlich benötigen, muß sie erneut eingegeben werden.

```
5 CALL CLEAR
```

eingeben und **ENTER** drücken.

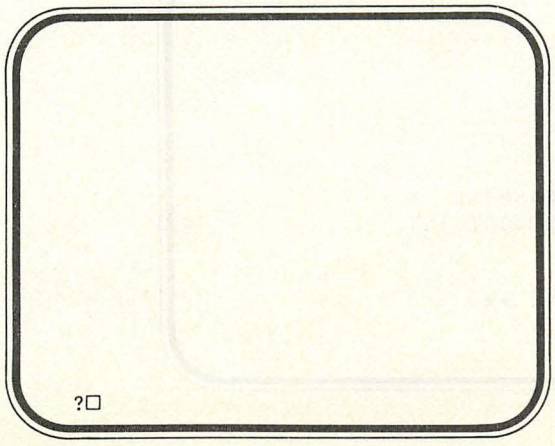
Die INPUT-Anweisung

Wie wir bereits gesehen haben, läßt sich durch einfache Neueingabe der Zeile 10 der Wert von K leicht ändern, da der Variablen damit ein neuer Wert zugeordnet wird. Aber angenommen, es liegen viele Werte für K vor und für jeden Wert soll der entsprechende P-Wert berechnet werden. Das wäre doch recht ermüdend, Zeile 10 jedesmal erneut eingeben zu müssen.

Es gibt eine weitaus bessere Methode, Zeile 10 zu editieren. Eine INPUT-Anweisung teilt dem Rechner mit, ein Fragezeichen auszugeben und zu stoppen; er wartet dann auf die Eingabe eines Wertes und das Drücken der **ENTER**-Taste. Der vom Anwender eingegebene Wert wird dann der in der INPUT-Anweisung enthaltenen Variablen zugeordnet. Als Beispiel soll

```
10 INPUT K
```

eingegeben und **ENTER** gedrückt werden. Nun läßt man das Programm erneut ablaufen.



Einfaches Programmieren

Das Fragezeichen und der Positionsanzeiger zeigen an, daß der Rechner darauf wartet, daß der Anwender einen Wert für K "eingibt" (INPUT). Jetzt soll K = 70 sein, daher wird die Ziffer 70 eingegeben und *ENTER* gedrückt. Der Computer druckt die Antwort aus:

```
? 70
DIE ANTWORT IST:
 154
** DONE **
>□
```

Nun kann das Programm so oft wie gewünscht ausgeführt werden, wobei der Wert von K jedesmal geändert werden kann, wenn der Rechner ein Fragezeichen ausdruckt und stoppt. Lassen Sie das Programm mehrere Male mit unterschiedlichen Werten für K laufen.

Die INPUT-Anweisung kann auch für das Ausdrucken einer "Aufforderungs" -Nachricht (anstelle eines Fragezeichens) verwendet werden, die den Anwender daran erinnert, auf welchen Wert der Rechner wartet. Zeile 10 wird erneut durch Eingabe von:

```
10 INPUT   "KILOGRAMM?":K
```

Doppelpunkt

und Drücken von *ENTER* geändert. Jetzt das Programm erneut ablaufen lassen. Zuerst fragt das Programm nach:

```
KILOGRAMM?
```

Dieses Mal soll K = 50 sein. Jetzt die Ziffer 50 tippen und *ENTER* drücken.

```
KILOGRAMM?50
DIE ANTWORT IST:
 110

** DONE **

>□
```

Einfaches Programmieren

Mittlerweile sieht das Programm folgendermaßen aus:

```
5 CALL CLEAR
10 INPUT "KILOGRAMM?":K
20 LET P=2.2*K
27 PRINT "DIE ANTWORT IST:"
30 PRINT P
40 END
```

Falls gewünscht kann das Programm zu diesem Zeitpunkt auch auf dem Bildschirm dargestellt werden, um die bisher gemachten Änderungen zu überprüfen. Danach nehmen wir eine weitere Änderung vor.

String-Variable

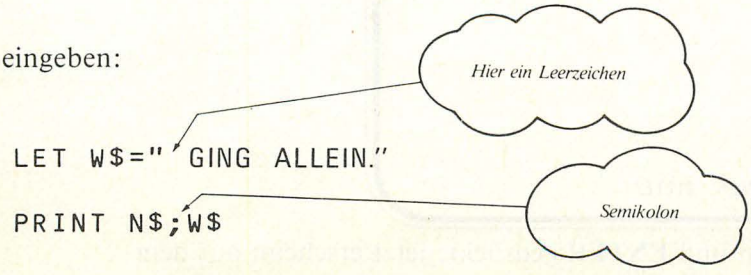
Die numerischen Variablen wurden bereits vorgestellt: Zahlenwerte, denen Namen (Variable) zugeordnet sind, wie zum Beispiel "K = 50". Eine String-Variable ist eine Kombination von Zeichen (Buchstaben und Ziffern oder andere Symbole), die einem Namen zugeordnet ist. String-Variable unterscheiden sich folgendermaßen von den numerischen:

1. Der Name der Variablen muß mit einem $ aufhören.(**SHIFT $**).
2. Die alphanumerischen Zeichen des "Strings" müssen zwischen Anführungszeichen stehen.
3. Strings können nicht in Rechenoperationen auftreten.

Nun werden zuerst ein paar Beispiele im Direktbetrieb ausprobiert, bevor das Programm geändert wird. (Dies hat keinerlei Auswirkung auf das im Speicher abgespeicherte Programm!) Den Bildschirm löschen (CALL CLEAR) und folgendes eingeben:

```
LET N$="HAENSCHEN KLEIN"

PRINT N$
```

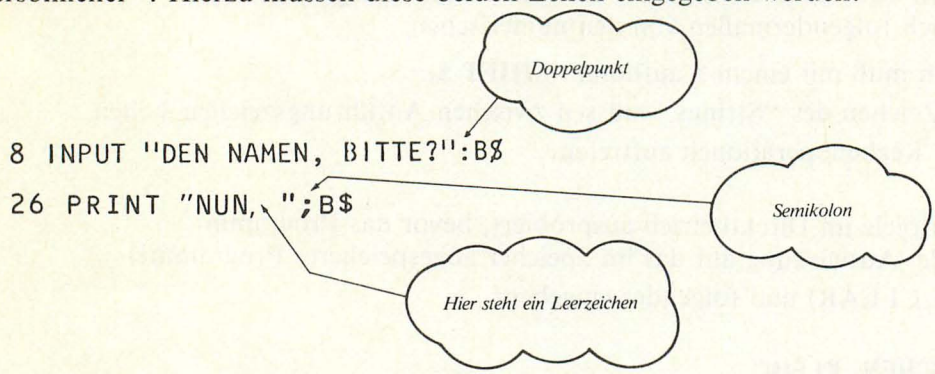
Jetzt dies eingeben:



Einfaches Programmieren

```
>LET N$="HAENSCHEN KLEIN"
>PRINT N$
 HAENSCHEN KLEIN
>LET W$="GING ALLEIN"
>PRINT N$;W$
 HAENSCHEN KLEIN GING ALLEIN
 >□
```

Durch die Verwendung einer String-Variablen wird unser Umwandlungsprogramm jetzt etwas "persönlicher". Hierzu müssen diese beiden Zeilen eingegeben werden:



(Den Bildschirm löschen und das Programm erneut auflisten, um zu sehen, wie die neuen Zeilen eingefügt wurden.)

Wenn das so geänderte Programm erneut läuft, halten die beiden INPUT-Anweisungen das Programm zweimal an:

Der Rechner fragt	Der Anwender gibt ein
DEN NAMEN, BITTE ? □	Den Namen und dann **ENTER** drücken.
KILOGRAMM? □	Die Anzahl der Kilogramm und dann **ENTER** drücken.

Probieren geht über studieren: Run eingeben und **ENTER** drücken.

```
DEN NAMEN, BITTE?□
```

Angenommen, es wird Alpha eingegeben und **ENTER** gedrückt; jetzt erscheint auf dem Bildschirm:

```
DEN NAMEN,BITTE?ALPHA
KILOGRAMM? □ .
```

Für die Anzahl der Kilogramm wird erneut 70 eingegeben. Wiederum **ENTER** drücken und auf dem Bildschirm sieht man:

```
DEN NAMEN,BITTE?ALPHA
KILOGRAMM?70
NUN, ALPHA
DIE ANTWORT IST
 154

** DONE **

>□
```

Wird eine Nachricht (beispielsweise ein Name oder eine Aufforderungsanweisung) mehr als einmal in einem Programm verwendet, können String-Variable viel unnötiges Tippen vermeiden.

Nun wird diese Programmversion aufgelistet, und die neuesten Änderungen werden überprüft. An dieser Stelle wäre es gut, selbständig etwas zu experimentieren, um einige der erlernten Dinge auszuprobieren.

Experiment!

Wie wäre es mit einer kleinen Aufgabe? Sie besteht darin, ein anderes Umwandlungsprogramm zu schreiben - hier wird die Temperatur von Grad Fahrenheit (F) in Grad Celsius (C) umgewandelt. Die Gleichung für die Umformung lautet

Grad C = 5/9 (Grad F – 32)

Nicht die INPUT-Anweisungen und CALL CLEAR an den notwendingen Stellen vergessen!
Hinweis: `C=5/9x(F-32)` setzen - die Klammern müssen auf jeden Fall im Programm stehen!

Einfaches Programmieren

Die GO TO-Anweisung

Bisher wurden Programme entwickelt, die von Anfang bis zum Ende in normaler, fortläufender Reihenfolge ablaufen. Doch gibt es viele Situationen, in denen dieser geordnete Fluß der Programmschritte unterbrochen werden muß. Sehen wir uns dazu das folgende Programm an, das jedoch noch nicht eingegeben wird:

```
10 CALL CLEAR
20 INPUT K
30 PRINT K
40 PRINT
50 K=K+1
60 GO TO 30
```

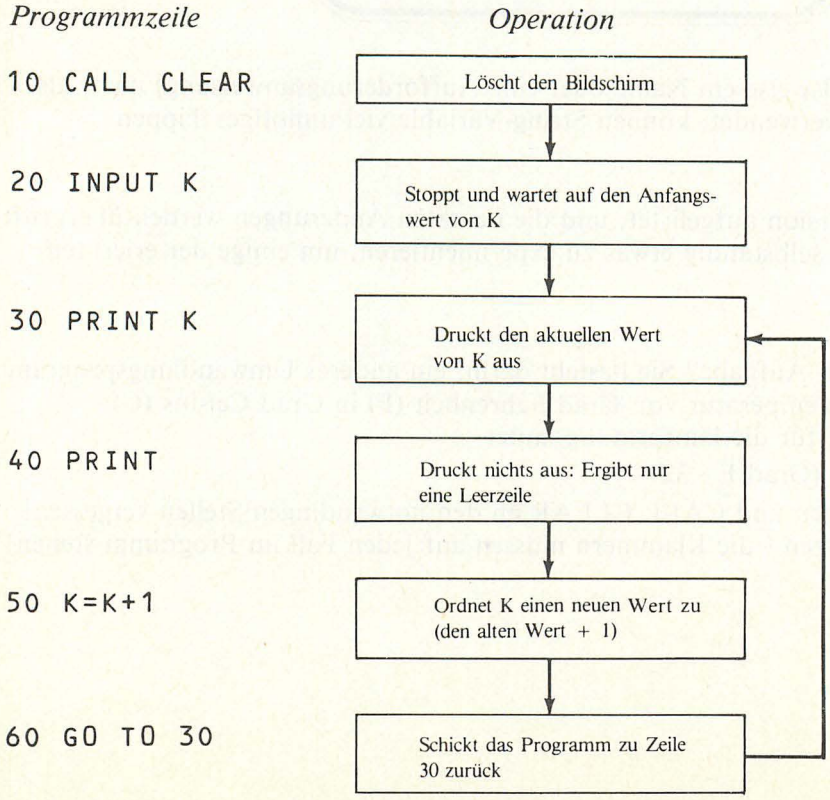
Hier wird das Programm über die GO TO-Anweisung der Zeile 60 zur Zeile 30 "zurückgeschickt". Die GO TO-Anweisung hat zur Folge, daß die von den Zeilen 30, 40 und 50 ausgeführten Schritte ständig wiederholt werden, es handelt sich hierbei um eine sogenannte Schleife. (Achtung: Im Programm erscheint keine END-Anweisung. Denn das Programm kommt niemals über Zeile 60 hinaus! Es läuft immer weiter, bis **CLEAR** gedrückt wird. Dies wird als "Endlosschleife" bezeichnet.)

Nun das Programm eingeben; zuerst NEW tippen und dann **ENTER** drücken, was den Speicher des Computers löscht. Danach die folgenden Zeilen eingeben:

```
10 CALL CLEAR
20 INPUT K
30 PRINT K
40 PRINT
50 K=K+1
60 GO TO 30
```

LET ist wahlweise

Vor dem Ablauf des Programms sei die nachstehende Abbildung betrachtet, ein sogenanntes Flußdiagramm; dieses erläutert die Arbeitsweise des Programms.



Einfaches Programmieren

Nun lassen wir das Programm laufen und geben 1 für den Anfangswert von K ein. Auf dem Bildschirm läßt sich verfolgen, wie schnell der Computer rechnet – fast zu schnell! Aus diesem Grunde wurde die Leerzeile (Zeile 40) eingeschoben. Dadurch werden die Zahlen ein wenig auseinandergezogen, damit sie besser zu sehen sind.

Lassen wir den Computer ruhig noch ein Weilchen rechnen. Um ihn zu stoppen, wird **CLEAR** gedrückt. Auf dem Bildschirm erscheint * BREAKPOINT AT (Zwischenstop bei), was anzeigt, wo das Programm angehalten wurde. Das Programm sollte ruhig so oft laufen wie gewünscht, und als Anfangswert für K kann jede beliebige Zahl dienen (50, 100, 500 etc.).

GO TO kann als GOTO im Programm erscheinen, denn der Computer ist da gar nicht kleinlich. Wird jedoch versucht, das Programm zu einer nicht existierenden Zeilennummer zu schicken, gibt er eine Fehlermeldung aus. Als Beispiel wird

```
60 GO TO 25
```

eingegeben und **ENTER** gedrückt. Dann versuchen wir, das Programm laufen zu lassen. Und was geschieht? Es erscheint die folgende Fehlermeldung:

```
* BAD LINE NUMBER IN 60
```

Daher wird die betreffende Zeile folgendermaßen abgeändert

```
60 GO TO 30
```

und das Programm erneut abgewickelt.

Kann das Programm so umgebaut werden, daß es in Zweier- oder Fünferschritten zählt? Sicher! Durch nur eine Änderung im Programm zählt der Computer in Zweierschritten:

```
50 K=K+2
```

eingeben und **ENTER** drücken. Jetzt wird das Programm zum Ablauf gebracht, und auf die Frage des Rechners nach dem Anfangswert von K wird 2 getippt.

Man sollte mit dem Programm ruhig noch etwas experimentieren, und es in Dreier-, Fünfer-, Zehnerschritten etc. zählen lassen.

Wenn Sie als Anfänger Ihre Programmiererfahrung erweitern wollen, um mit dem Home-Computer noch vertrauter zu werden, dann empfehlen wir Ihnen das Buch "BASIC für Anfänger".

Dieses Buch vermittelt Ihnen in unterhaltsamer Form und in kurzer Zeit die ersten Programmier-Kenntnisse in TI-BASIC im Selbststudium.

TI-BASIC Befehle und Statements

Einführung

Dieser Abschnitt der Bedienungsanleitung enthält eine vollständige Erklärung aller Befehle und Statements. Sie sind Bestandteil der TI-BASIC-Sprache und direkt in den Home-Computer integriert. Wie bereits erwähnt, ist BASIC eine Computersprache, die auch für den Anfänger sehr anwenderfreundlich konzipiert ist, die aber dennoch durch ihre Leistungsfähigkeit die Anwendung des Computers für einen weiten Anwendungsbereich möglich macht.

Wenn Sie schon einige Programmierkenntnisse haben und lediglich mit dem TI-BASIC und seiner Funktion in Verbindung mit dem TI-Computer vertraut werden wollen, haben wir eine Reihe von Anwenderprogrammen am Ende dieses Buchs vorbereitet. Diese Programme beginnen auf einem sehr einfachen Niveau und werden dann zu größerer Komplexität gesteigert. Die Beschäftigung mit diesen Programmen veranschaulicht Ihnen die Anwendung von vielen Anweisungen des TI-BASIC. Bei Fragen gibt Ihnen die vorliegende Bedienungsanleitung detaillierte Informationen.

Wenn Sie zwar Programmiererfahrung haben, aber noch nicht in BASIC programmiert haben, oder wenn Sie vor dem Benutzen Ihres Computers nur Ihre Kenntnisse auffrischen wollen, empfehlen wir zur Einführung das ausgezeichnete Buch von Herbert Peckam, "Programming BASIC with the TI Home Computer".

Es vermittelt schnelle, anspruchsvolle BASIC-Erfahrungen, und ist über den Buchhandel zu beziehen.
Wenn Sie detaillierte Kenntnisse besitzen oder schon Programmierexperte sind, soll Ihnen dieses Buch primär als Nachschlagewerk für die Programmier-Befehle des TI-BASIC dienen und Ihnen diejenigen Details liefern, die von Zeit zu Zeit aufgefrischt werden müssen. Das TI-BASIC stimmt mit dem American National Standard for Minimal BASIC überein. Weitere Merkmale in TI-BASIC wie Farbgrafiken und akustische Signale etc. werden ebenfalls beschrieben.

Wenn Sie ein erfahrener BASIC-Programmierer sind, werden Sie keinerlei Schwierigkeiten haben.

Gliederung dieser Anleitung

Diese Anleitung ist nach Tastenfunktionen und nach folgenden Funktionsgruppen gegliedert:

1. Allgemeine Informationen
2. COMMANDS: Unmittelbar ausführbare Befehle
3. STATEMENTS: Programmbefehle
4. Ein-/Ausgabe-Anweisungen
5. Farbgraphiken und akustische Signale
6. Eingebaute numerische Funktionen
7. Eingebaute String-Funktionen
8. Benutzerdefinierte Funktionen
9. Datenfelder
10. Unterprogramme (Subroutinen)
11. Datenverarbeitung

Vereinbarungen zur Schreibweise

Zu Beginn der Erläuterung jedes einzelnen Befehls und jedes Statements in TI-BASIC erscheint eine Zeile, die das allgemeine Format für die Eingabe des Befehls oder Statements angibt. Bei diesen Formatzeilen wurden bestimmte Vereinbarungen bezüglich der Schreibweise eingehalten.

{ } – Die geschweiften Klammern weisen darauf hin, daß Sie die Wahl haben, was Sie verwenden. Sie dürfen aber nur eine der innerhalb der Klammern angegebenen Option (Wählmöglichkeit) verwenden.

[ ] – Die eckige Klammer gibt an, daß es Ihnen freisteht, die Option (Wählmöglichkeit) innerhalb der Klammer zu benutzen. Sie können sie zwar anwenden, aber es ist nicht erforderlich.

... – Die Punkte besagen, daß Sie den vorangehenden Ausdruck beliebig oft wiederholen können.

GROSSBUCHSTABEN – Wörter in Großbuchstaben müssen genau nach den Angaben eingetippt werden, wenn Sie sie anwenden.

Kursivdruck – Wörter in Kursivdruck sind eine allgemeine Beschreibung des Ausdruckes, der hier erscheinen muß. Sie müssen also an diese Stelle den von Ihnen selbst gewählten Ausdruck setzen.

Beispiele

Für jede Anweisung und für jeden Befehl werden in dieser Anleitung am rechten Rand jeder Seite Programmbeispiele aufgeführt. Jede Zeile, die Sie eingeben müssen, wird durch das Dialogzeichen > links von der Zeile angedeutet, so, wie es am Bildschirm erscheint.

Allgemeine Informationen

Einführung

Ist Ihr Computer aufgestellt und in Betrieb genommen, so können Sie sofort mit TI-BASIC beginnen. Beim Einschalten Ihres Computers erscheint auf dem Schirm das Standardbild. Drücken Sie dann eine beliebige Taste, um die Hauptwahlliste auf den Bildschirm zu rufen. Nun drücken Sie 1. Damit wählen Sie TI-BASIC. Mit Ausnahme der Worte "TI BASIC READY" (TI BASIC bereit) und einem Dialogzeichen, gefolgt von einem blinkenden Positionszeiger (■), ist der Bildschirm leer. Immer, wenn der Positionszeiger auf dem Bildschirm erscheint, erwartet der Computer Ihre Eingabe. Das Dialogzeichen > markiert den Beginn jeder Zeile, die Sie eintippen.

Jede Zeile des Bildschirms kann bis zu 28 Zeichen anzeigen. Die Länge eines Statements umfaßt maximal vier Bildschirmzeilen. Wenn eine Bildschirmzeile voll ist, geht der Positionszeiger automatisch auf die nächste Zeile herunter, wenn Sie weitertippen. Sind vier Zeilen vollgeschrieben, akzeptiert der Computer zwar weitere Zeichen, aber der Positionszeiger bleibt an der gleichen Stelle. Jedes Zeichen, das Sie dann eingeben, ersetzt das letzte Zeichen der Zeile.

Alle Tasten, die in dem Abschnitt "Spezielle Tastenfunktionen" behandelt wurden, können vor Drücken der Taste **[ENTER]** zur Modifizierung (Editieren) der Programmzeilen verwendet werden.

Um eine Programmzeile nach Drücken von **[ENTER]** zu ändern, tippen Sie entweder die gesamte Programmzeile neu ein und nehmen dabei die gewünschten Korrekturen vor, oder Sie geben den EDIT-Modus ein. Beachten Sie, daß alle offenen Dateien abgeschlossen werden (siehe OPEN-Statement), wenn Sie über den EDIT-Modus das Programm in irgendeiner Form modifizieren; alle Variablen werden undefiniert.

Spezielle Tastenfunktionen

Mehrere Tasten haben in TI-BASIC spezielle Funktionen.

[ENTER] – Beim Drücken der Taste **[ENTER]** übernimmt der Computer die eben eingetippte Programmzeile (max. 4 Bildschirmzeilen).

[FCTN =] (QUIT) – Beim Drücken der Taste QUIT kehrt der Computer zum Standardbild am Schirm zurück. Wenn der Computer den TI-BASIC-Modus verläßt, werden alle gespeicherten Daten und das Programm gelöscht.

[FCTN ↑] (UP) – Die Taste mit dem nach oben gerichteten Pfeil funktioniert genau wie die **[Enter]**-Taste, ausgenommen im EDIT-Modus.

[FCTN ↓] (DOWN) – Die Taste mit dem nach unten gerichteten Pfeil funktioniert genau wie die **[ENTER]**-Taste, ausgenommen im EDIT-Modus.

[FCTN ←] (LEFT) – Die Taste mit dem linksgerichteten Pfeil (Rücktaste) bewegt den Positionszeiger bei jedem Drücken um eine Stelle nach links. Wenn der Positionszeiger über ein Zeichen läuft, wird es weder gelöscht noch verändert. Erreicht der Positionszeiger den Zeilenanfang, hat das Drücken der Rücktaste keine Wirkung.

[FCTN →] (RIGHT) – Die Taste mit dem rechtsgerichteten Pfeil bewegt den Positionszeiger bei jedem Drücken um eine Stelle nach rechts. Mit dieser Taste können Sie den Positionszeiger über ein Zeichen führen, ohne es zu löschen oder zu verändern. Erreicht der Positionszeiger das Programm-Zeilenende (4 Bildschirmzeilen), hat das Drücken der Vorwärtstaste keine Wirkung.

[FCTN 2] (INS) – Die INSERT-Taste verwendet man, um in der Mitte einer Programmzeile Zeichen einzufügen. Zum Einfügen von Zeichen bringen Sie (mit den Tasten **[FCTN ←]** oder **[FCTN →]**) den Positionszeiger unmittelbar auf das Zeichen rechts von der Stelle, wo Sie das Zeichen ergänzen wollen und drücken dann die INSERT-Taste.

Jedesmal, wenn Sie im Anschluß an INSERT ein Zeichen eingeben, werden der Positionszeiger und jedes Zeichen der Programmzeile, das sich nicht links vom Positionszeiger befindet, um eine Stelle nach rechts bewegt. Das Zeichen für die von Ihnen gedrückte Taste wird dann in die leere Stelle eingeschoben, die durch die Bewegung des Positionszeigers und der anderen Zeichen entstanden ist. Beachten Sie, daß Zeichen, die über das Ende der Programmzeile hinausgeschoben werden, gelöscht werden.

Spezielle Tastenfunktionen

[FCTN 1] (DEL) – Die Taste DELETE verwendet man zum Löschen von Zeichen aus einer Programmzeile. Um Zeichen zu löschen, bringt man (mit den Tasten **[FCTN ←]** und **[FCTN →]**) den Positionszeiger auf das Zeichen, das entfernt werden soll, und drückt dann die DELETE-Taste. Dabei wird das Zeichen unter dem Positionszeiger gelöscht, und alle Zeichen der Programmzeile rechts vom Positionszeiger verschieben sich um eine Stelle nach links.

[FCTN 4] (CLEAR) – Die CLEAR- oder BREAK-Taste hat zwei Funktionen, je nachdem, wann sie verwendet wird:

- Drückt man diese Taste, während ein Programm abläuft, erfolgt bei der Programmzeile, die als nächste durchzuführen ist, eine Programmunterbrechung. Diese Taste gibt Ihnen also die Möglichkeit, ein Programm während des Ablaufs vorübergehend zu stoppen.
  Beachten Sie, daß Sie die Break-Taste gedrückt halten müssen, bis das Programm den Ablauf unterbricht. Wenn der Programmlauf mit der Break-Taste unterbrochen wird, wird die Nachricht "BREAKPOINT AT Zeilennr." angezeigt.
  Die mit der Zeilennummer angegebene Programmzeile wird dann nicht durchgeführt. Sie können mit dem CONTINUE-Befehl den Programmablauf dort wieder starten, wo er unterbrochen wurde.
- Drückt man die Clear-Taste während des Eintippens einer Programmzeile, wird diese gelöscht. Die Taste hat noch weitere Funktionen im EDIT-Modus und im NUMBER-MODE.

[FCTN 3] (ERASE) – Die Taste ERASE löscht die gesamte Programmzeile, die Sie gerade eintippen. Die Zeile wird nicht eingegeben.

LEERTASTE – Bei jedem Drücken bewegt die Leertaste den Positionszeiger um eine Stelle nach rechts. Wenn Sie mit der Leertaste den Positionszeiger über ein Zeichen führen, wird dieses durch eine Leerstelle ersetzt.

Leerstellen

Im allgemeinen kann eine Leerstelle fast überall im Programm vorkommen, ohne dessen Durchführung zu beeinflussen. Alle zusätzlich eingebauten, aber nicht erforderlichen Leerstellen werden jedoch gelöscht, wenn die Programmzeile durch den EDIT-, NUM- oder LIST-Befehl angezeigt wird. Es gibt einige Ausnahmen, wo Leerstellen nicht auftreten dürfen, insbesondere in folgenden Fällen:

(1) innerhalb einer Zeilennummer
(2) innerhalb eines reservierten Worts
(3) innerhalb einer numerischen Konstanten
(4) innerhalb einer Variablenbezeichnung

Nachstehend sehen Sie einige Beispiele für eine unkorrekte Anwendung von Leerstellen. Die korrekte Zeile erscheint in der rechten Spalte:

(1) 1 00 PRINT "HELLO"	`>100 PRINT "HELLO"`
(2) 110 PR INT "HOW ARE YOU?"	`>110 PRINT "HOW ARE YOU?"`
(3) 120 LET A = 10 0	`>120 LET A=100`
(4) 130 LET CO ST = 24.95	`>130 LET COST=24.95`

Allen reservierten Wörtern muß eines der nachstehenden Elemente unmittelbar vorangehen oder folgen:

- eine Leerstelle
- ein arithmetischer Operator (+, –, *, /,)
- der String-Operator (&)
- ein Spezialzeichen, das in einem bestimmten Statementformat verwendet wird (< = > ( ). ; : #)
- Ende der Zeile (**[ENTER]** -Taste)

Zeilennummern

Jedes Programm besteht aus einer Folge von Programmzeilen. Die Zeilennummer dient als Markierung für die Programmzeile. Jede Zeile im Programm beginnt mit einer Zeilennummer, die eine ganze Zahl im Bereich 1 bis 32767 inkl. sein muß. Führende Nullen kann man zwar verwenden, sie werden aber vom Computer ignoriert.

Beispiel: 033 und 33 werden in jedem Fall als 33 gelesen. Es ist nicht nötig, die Zeilen in fortlaufender Reihenfolge einzugeben; sie werden automatisch vom Computer sortiert.

Wenn Sie das Programm ablaufen lassen, werden die Programmzeilen in aufsteigender Reihenfolge durchgeführt, bis einer der folgenden Punkte wirksam wird:

(1) ein Verzweigungsbefehl wird durchgeführt (siehe "Allgemeine Programm-Statements")
(2) bedingt durch einen auftretenden Fehler wird der Programmablauf unterbrochen
(3) der Anwender unterbricht den Programmablauf selbst mit einem BREAK-Befehl oder mit der Break-Taste **[FCTN 4]** (CLEAR)
(4) ein STOP-Statement oder ein END-Statement wird durchgeführt
(5) das Statement mit der höchsten Zeilennummer wurde durchgeführt

Wenn Sie eine Programmzeile mit einer Zeilennummer kleiner als 1 oder größer als 32767 eingeben, wird die Fehlernachricht "BAD LINE NUMBER" (falsche Zeilennummer) angezeigt, und die Zeile wird nicht in den Speicher eingegeben.

```
>0 A=2

 * BAD LINE NUMBER

>33000 C=4

 * BAD LINE NUMBER
```

Numerische Konstanten

Die numerischen Konstanten müssen entweder positive oder negative reelle Zahlen sein. Sie können die numerischen Konstanten mit jeder beliebigen Stellenzahl eingeben. Die Werte werden intern in Radixschreibweise für ein System mit der Basis 7 gespeichert. Das heißt, die Zahlen haben 13 oder 14 Dezimalstellen, abhängig vom Wert der Zahl.

Beispiele:

```
>PRINT 1.2
 1.2

>PRINT -3
 -3

>PRINT 0
 0
```

Exponentialform

Sehr große oder sehr kleine Zahlen werden in Exponentialform verarbeitet. Eine Zahl in Exponentialform ist als Basiszahl (Mantisse), multipliziert mit einer Zehnerpotenz (Exponent), ausgedrückt.
Zahl = Mantisse $\times$ 10^{Exponent}

Eingabe einer Zahl in Exponentialform:

Zuerst geben Sie die Mantissen ein. (Achten Sie darauf, das Minuszeichen zuerst einzugeben, wenn die Mantisse negativ ist). Dann tippen Sie den Buchstaben "E". (E muß großgeschrieben werden.) Anschließend wird die Potenz von 10 eingegeben.
(Ist sie negativ, achten Sie auf das Minuszeichen vor der Eingabe des Exponenten).

Zahl	Eingabeform
3.264×10^{4}	3.264E4
-98.77×10^{21}	—98.77E21 oder —9.877E22
5.691×10^{-5}	5.691E—5
-2.47×10^{-17}	—2.47E—17

```
>PRINT 3.264E4
 32640

>PRINT -98.77E21
 -9.877E+22
```

Numerische Konstanten sind im Bereich von —9.9999999999999E127 bis —1E—128, 0 und von 1E—128 bis 9.9999999999999E127 definiert.

Bereichsunterschreitung

Ist eine eingegebene oder berechnete Zahl nach der Rundung größer als —1E—128 und kleiner als 1E—128, liegt eine Bereichsunterschreitung vor. In diesem Fall ersetzt der Computer den Wert der Zahl durch eine Null und der Programmablauf wird fortgesetzt.

Eine Warnung oder Fehlermeldung wird nicht gegeben.

```
>PRINT 0
 0

>PRINT -9E-130
 0

>PRINT 9E-142
 0
```

Kapazitätsüberlauf

Wird eine Zahl eingegeben oder berechnet, deren Wert nach der Rundung größer als 9.9999999999999E127 oder kleiner als —9.9999999999999E127 ist, liegt ein Kapazitätsüberlauf vor. In diesem Fall wird die Konstante durch den Grenzwert des Computers ersetzt, eine Warnung erfolgt mit der Nachricht "NUMBER TOO BIG" (zu große Zahl), und der Programmablauf wird fortgesetzt. Die Computergrenze liegt bei —9.9999999999999E127 oder 9.9999999999999E127.

Beachten Sie, daß das Symbol "**" gedruckt wird, wenn der Exponent größer als 99 ist.

```
>PRINT 97E136

 * WARNING:
   NUMBER TOO BIG
  9.99999E+**

>PRINT -108E144

 * WARNING:
   NUMBER TOO BIG
 -9.99999E+**
```

String-Konstanten

Unter einer String-Konstanten versteht man eine Folge von Zeichen (einschließlich Buchstaben, Zahlen, Leerstellen, Sondersymbole etc.), die in Anführungszeichen gesetzt sind. Leerstellen innerhalb der String-Konstanten werden nicht ignoriert und als Zeichen in der Folge mitgezählt. Alle Zeichen auf der Tastatur, die angezeigt werden können, dürfen auch in einer String-Konstanten verwendet werden. Eine String-Konstante ist durch die Länge der Eingabezeile limitiert (112 Zeichen oder vier Zeilen auf dem Bildschirm).

Beispiele:

```
>NEW

>100 PRINT "HI!"
>110 PRINT "THIS IS A STRING
 CONSTANT."
>120 PRINT "ALL CHARACTERS (+
 -*/ @,) MAY BE USED."
>130 END
>RUN
 HI!
 THIS IS A STRING CONSTANT.
 ALL CHARACTERS (+-*/ @,) MAY
 BE USED.

 ** DONE **
```

Wird ein PRINT- oder DISPLAY-Statement durchgeführt, werden die Anführungszeichen nicht angezeigt. Auf Wunsch können Wörter oder Sätze innerhalb eines Strings ebenfalls mit Anführungszeichen gedruckt werden; Sie fügen einfach die entsprechenden Anführungszeichen auf jeder Seite des entsprechenden Wortes oder Satzes ein, wenn Sie die String-Konstante eintippen.

```
>NEW

>100 PRINT "TO PRINT ""QUOTE
 MARKS"" YOU MUST USE DOUBLE
 QUOTES."
>110 PRINT
>120 PRINT "TOM SAID, ""HI, M
 ARY!"""
>130 END
>RUN
 TO PRINT "QUOTE MARKS" YOU M
 UST USE DOUBLE QUOTES.

 TOM SAID, "HI, MARY!"

 ** DONE **
```

String-Ausdrücke werden aus String-Variablen, String-Konstanten und Funktionsaufrufen mit der Verkettungsoperation (&) gebildet. Die Verkettungsoperation ermöglicht die Kombination von Strings. Alle in einem String-Ausdruck angegebenen Funktionen müssen entweder Funktionen des TI-BASIC sein (siehe eingebaute String-Funktion), oder durch ein DEF-Statement definiert werden, und sie müssen einen String-Wert haben. Wenn die Verarbeitung eines Stringausdrucks zu einem Wert führt, der die maximale String-Länge von 225 Zeichen übersteigt, wird der String auf der rechten Seite abgeschnitten, und das Programm läuft ohne Warnung weiter.

Beachten Sie, daß alle Zeichen in einem String immer genau so in der Anzeige erscheinen, wie sie eingegeben werden.

Beispiele:

```
>NEW

>100 A$="HI"
>110 B$="HELLO THERE!"
>120 C$="HOW ARE YOU?"
>130 MSG$=A$&SEG$(B$,6,7)
>140 PRINT MSG$&" "&C$
>150 END
>RUN
 HI THERE! HOW ARE YOU?

 ** DONE **
```

Variable

In BASIC erhalten alle Variablen eine Bezeichnung, einen Namen. Jeder Variablenname kann aus einem oder mehreren Zeichen bestehen, aber er muß mit einem Buchstaben, einem "at"-Zeichen (@ - kommerzielles zu je), einer linken Klammer ([ ), einer rechten Klammer ( ]), einem Schrägstrich (/) oder einem Unterstrich (_) beginnen. Die einzigen zulässigen Zeichen in einem Variablennamen sind Buchstaben, Zahlen, das Zeichen (@), und das Zeilenzeichen (__). Eine Ausnahme bildet das Dollarzeichen ($). Das letzte Zeichen einer STRING-Variablen *muß* das Dollarzeichen sein, und dies ist auch die einzige Stelle in einem Variablenamen, wo man es verwenden kann.
Variablennamen sind auf 15 Zeichen inkl. Dollarzeichen bei Namen für STRING-Variable limitiert.

Bei den Bezeichnungen für Datenfelder (arrays) gelten dieselben Regeln wie bei einfachen Variablennamen. (Weitere Informationen finden Sie im Abschnitt über Datenfelder). In einem einzigen Programm darf der gleiche Name nicht zweimal verwendet werden, einmal für einen einfachen Variablennamen und ein zweites Mal als Bezeichnung für ein Datenfeld, noch dürfen zwei Datenfelder mit unterschiedlichen Dimensionen die gleiche Bezeichnung tragen.

Zum Beispiel können Z und Z(3) nicht zusammen als Namen im gleichen Programm verwendet werden, ebensowenig X(3, 4) und X(2, 1, 3). Es gibt jedoch keine Beziehung zwischen dem Namen für eine numerische- und eine STRING-Variable, wo bis auf das Dollarzeichen völlige Übereinstimmung besteht. (X und X$ können in ein und demselben Programm verwendet werden).

Namen für numerische Variable
gültig: X, A9, ALPHA?, BASEPAY, V(3), T(X,3), TABELLE(X,XX7Y/2)
ungültig: X$, X/8, 3Y

Namen für STRING-Variable
gültig: S$, YZ2$, NAME$, Q5$ (3,X)
ungültig: S$3, X9, 4Z$

Wenn Sie einen Variablennamen mit mehr als 15 Zeichen eingeben, wird die Nachricht "BAD NAME" (falscher Name) angezeigt, und die Zeile wird nicht in den Speicher eingegeben.

Reservierte Wörter sind als Namen für Variable nicht zulässig, können aber als Teil der Variablenbezeichnung verwendet werden. LIST ist zum Beispiel kein gültiger Variablenname, dagegen wird LIST$ akzeptiert.

Während ein Programm abläuft, hat eine Variable zu jeder Zeit einen eindeutigen Wert. Wenn der Programmablauf beginnt, wird der jeder numerischen Variablen zugewiesene Wert auf Null gesetzt, und jede String-Variable erhält ebenfalls einen Ausgangswert von Null (d.h., ein String mit einer Länge von null Zeichen). Beim Ablauf des Programms werden den Variablen Werte zugeordnet, wenn LET-Statements, READ-Statements, FOR-TO-STEP-Statements oder INPUT-Statements durchgeführt werden.
Die Länge des STRINGS, dem eine Stringvariable zugeordnet wird, kann während des Programmlaufs zwischen 0 und 225 Zeichen variieren.

Beispiele:

```
>110 ABCDEFGHIJKLMNOPQ=3

 * BAD NAME
```

Reservierte Wörter

Unter reservierten Wörtern versteht man solche, die in TI-BASIC nicht als Variablenbezeichnungen verwendet werden dürfen. Beachten Sie, daß nur exakt das unten dargestellte Wort reserviert ist. Reservierte Wörter können Teil einer Variablenbezeichnung sein (ALEN und LENGTH sind zum Beispiel erlaubt). Nachstehend erhalten Sie eine vollständige Liste aller reservierten Wörter in TI-BASIC:

ABS
APPEND
ASC
ATN
BASE
BREAK
BYE
CALL
CHR$
CLOSE
CON
CONTINUE
COS
DATA
DEF
DELETE
DIM
DISPLAY
EDIT
ELSE
END
EOF
EXP
FIXED
FOR
GO
GOSUB
GOTO
IF
INPUT
INT
INTERNAL
LEN
LET
LIST
LOG
NEW
NEXT
NUM
NUMBER
OLD
ON
OPEN
OPTION
OUTPUT
PERMANENT
POS
PRINT
RANDOMIZE
READ
REC
RELATIVE
REM
RES
RESEQUENCE
RESTORE
RETURN
RND
RUN
SAVE
SEG$
SEQUENTIAL
SGN
SIN
SQR
STEP
STOP
STR$
SUB
TAB
TAN
THEN
TO
TRACE
UNBREAK
UNTRACE
UPDATE
VAL
VARIABLE

Numerische Ausdrücke

Numerische Ausdrücke werden aus numerischen Variablen, numerischen Konstanten und Funktionshinweisen mit arithmetischen Operatoren (+, —, *, /, ∧) gebildet. Alle Funktionen, die in einem Ausdruck angegeben sind, müssen entweder in TI-BASIC vertreten sein (siehe die Abschnitte über eingebaute Funktionen) oder durch ein DEF-Statement definiert werden (siehe Seite 103). Die beiden Arten von arithmetischen Operatoren (Präfix- und Infix-Operatoren) werden nachstehend erläutert.

Die arithmetischen Präfix-Operatoren sind Plus (+) und Minus (—), und sie werden zur Angabe des Vorzeichens (positiv oder negativ) bei Konstanten und Variablen verwendet. Das Pluszeichen bedeutet, daß die Zahl nach dem Präfix-Operator (+) mit +1 multipliziert werden muß, das Minuszeichen besagt, daß die Zahl nach dem Präfix-Operator (—) mit —1 multipliziert wird. Bei fehlendem Präfix-Operator wird die Zahl so behandelt, als wäre der Präfix-Operator plus. Hier einige Beispiele von Präfix-Operatoren mit Konstanten und Variablen:

10 —6 +3
+A —W

Die arithmetischen Infix-Operatoren werden für Berechnungen verwendet. Zu ihnen gehören: Addition (+), Subtraktion (—), Multiplikation (*), Division (/) und Potenzierung (∧). In einem numerischen Ausdruck muß zwischen jeder numerischen Konstanten und/oder Variablen ein Infix-Operator stehen. Beachten Sie, daß eine Multiplikation nicht durch einfaches Aneinandersetzen der Variablen oder durch Anwendung von Klammern angegeben werden kann. Sie müssen den Multiplikations-Operator (*) anwenden.

Infix- und Präfix-Operatoren können innerhalb eines numerischen Ausdrucks nebeneinander eingegeben werden. Die Operatoren werden in der üblichen Weise ausgewertet.

Bei den Berechnungen der numerischen Ausdrücke werden auch in TI-BASIC die Standardregeln der mathematischen Hierarchie angewandt. Diese Regeln sind hier noch einmal aufgeführt:

1. Alle Ausdrücke innerhalb von Klammern werden entsprechend den Regeln der Hierarchie zuerst berechnet.
2. In der Reihenfolge von links nach rechts werden als nächstes Potenzierungen durchgeführt.
3. Die Präfix-Operatoren plus und minus werden verarbeitet.
4. Anschließend werden Multiplikationen und Divisionen abgeschlossen.
5. Dann folgen Additionen und Subtraktionen.

Beachten Sie, daß 0 ∧ 0 nach dem üblichen mathematischen Usus als 1 definiert ist.

Beispiele:

```
>NEW

>100 A=6
>110 B=4
>120 C=20
>130 D=2
>140 PRINT A*B/2
>150 PRINT C-D*3+6
>160 END
>RUN
  12
  20

 ** DONE **
```

```
>NEW

>100 A=2
>110 B=3
>120 C=4
>130 PRINT A*(B+2)
>140 PRINT B^A-4
>150 PRINT -C^A;(-C)^A
>160 PRINT 10-B*C/6
>170 END
>RUN
  10
  5.
 -16  16
  8

 ** DONE **
```

```
>PRINT 0^0
  1
```

Vergleiche und logische Variable

Im allgemeinen benutzt man logische Variable im IF-THEN-ELSE-Statement. Sie können aber überall verwendet werden, wo numerische Ausdrücke zulässig sind. Bei Verwendung von Vergleichen innerhalb numerischer Ausdrücke erhält die logische Variable den Wert —1, wenn die Relation zutrifft, und den Wert 0, wenn die Relation nicht zutrifft.

Vergleiche werden von links nach rechts vor der String-Verkettung und nach Abschluß aller arithmetischen Operationen innerhalb des Ausdrucks durchgeführt. Um die Stringverkettung vor den Vergleichen und/oder Vergleiche vor arithmetischen Operationen durchzuführen, müssen Klammern gesetzt werden. Gültige Vergleichsoperatoren sind:

- gleich (=)
- kleiner als (<)
- größer als (>)
- ungleich (< >)
- kleiner oder gleich (< =)
- größer oder gleich (> =)

Eine Erklärung, wie String-Vergleiche durchgeführt werden, um ein richtiges oder falsches Resultat zu erhalten, finden Sie beim Statement IF-THEN-ELSE. Bedenken Sie daß das Resultat, das Sie aus der Auswertung eines Vergleichsoperators erhalten, immer eine Zahl ist. Wenn Sie versuchen, das Resultat als String zu verwenden, verursachen Sie einen Fehler.

Beispiele:

```
>NEW

>100 A=2<5
>110 B=3<=2
>120 PRINT A;B
>130 END
>RUN
 -1  0

 ** DONE **
```

```
>NEW

>100 A$="HI"
>110 B$=" THERE!"
>120 PRINT (A$&B$)="HI!"
>130 END
>RUN
  0

 ** DONE **

>120 PRINT (A$&B$)>"HI"
>RUN
 -1

 ** DONE **

>120 PRINT (A$>B$)*4
>RUN
 -4

 ** DONE **
```

```
>NEW

>100 A=2<4*3
>110 B=A=0
>120 PRINT A;B
>130 END
>RUN
 -1  0

 ** DONE **
```

COMMANDS (unmittelbar ausführbare Befehle)

Einführung

Wann immer das Dialogzeichen und der blinkende Positionszeiger (□) im unteren Teil des Bildschirms erscheinen, befindet sich Ihr Computer im COMMAND MODE (Direktmodus). Befehle können dann ohne Voranstellen einer Zeilennummer eingegeben werden. Wird ein Befehl eingegeben, führt der Computer die Aufgabe sofort aus. Die einzelnen COMMANDS werden nun besprochen.

NEW

NEW

Der NEW-Befehl löscht das gegenwärtig gespeicherte Programm. Die Eingabe des NEW-Befehls hebt die Wirkung des BREAK-Befehls und die des TRACE-Befehls auf.

Der NEW-Befehl schließt auch alle offenen Dateien ab (siehe OPEN-Statement, und gibt alle Leerstellen frei, die mit Spezialzeichen belegt waren.

Zusätzlich löscht der NEW-Befehl alle Variablenwerte sowie die Tabelle, in der die Variablennamen gespeichert sind. Mit dem NEW-Befehl wird der Bildschirm gelöscht, und die Nachricht "TI BASIC READY" (TI-BASIC bereit) wird angezeigt. Das Dialogzeichen und der blinkende Positionszeiger (■) weisen darauf hin, daß Sie nun neue Befehle oder Programmzeilen eingeben können.

Beispiele:

```
TI BASIC READY

>□
```

LIST

List [*Liste der Zeilennummern*]
"Geräte-Name" [*: 3. Liste der Zeilennummern*]

Wenn der LIST-Befehl eingegeben wird, werden die durch die Liste der Zeilennummern spezifizierten Programmzeilen aufgelistet. Wird die Gerätebezeichnung eingegeben, werden die spezifizierten Programmzeilen auf diesem Gerät ausgegeben. Die Gerätebezeichnung für mögliche künftige Peripheriegeräte wird in den entsprechenden Bedienungsanleitungen genannt. Wenn man keine Gerätebezeichnung eingibt, werden die spezifizierten Zeilen auf dem Bildschirm angezeigt.
Gibt man den LIST-Befehl ohne Liste der Zeilennummern ein, wird das gesamte Programm aufgelistet. Die Programmzeilen werden immer in ansteigender Folge aufgelistet.

Beispiele:

```
>NEW

>100 A=279.3
>120 PRINT A;B
>110 B=-456.8
>130 END
>LIST
 100 A=279.3
 110 B=-456.8
 120 PRINT A;B
 130 END
```

LIST

Die Liste der Zeilennummern kann aus einer Einzelzahl, einer Einzelzahl mit vorangestelltem Bindestrich (zum Beispiel: −10), einer Einzelzahl mit nachgestelltem Bindestrich (zum Beispiel: 10−), oder einem durch Bindestrich fixierten Bereich von Zeilennummern bestehen.

- **Einzelzahl** – nur die Programmzeile für die angegebene Zeilennummer erscheint.
- **Einzelzahl mit vorangestelltem Bindestrich** – alle Programmzeilen mit Zeilennummern kleiner oder gleich der angegebenen Zeilennummer erscheinen.
- **Einzelzahl mit nachgestelltem Bindestrich** – alle Programmzeilen mit Nummern größer oder gleich der angegebenen Zeilennummer erscheinen.
- **Durch Bindestrich fixierter Zeilennummern-Bereich** – alle Programmzeilen werden angezeigt, deren Nummern nicht kleiner als die erste Zeilennummer und nicht größer als die zweite Zeilennummer sind.

Wenn ein Programm im Speicher ist, aber keine Programmzeilen innerhalb des spezifizierten Bereichs existieren, wird eine Programmzeile nach den folgenden Regeln angezeigt. Die Zeilenliste gibt an:

- **Zeilennummern größer als alle Zeilennummern des Programms** – die Programmzeile mit der höchsten Nummer wird angezeigt.
- **Zeilennummern kleiner als alle Zeilennummern des Programms** – die Programmzeile mit der niedrigsten Nummer wird angezeigt.
- **Zeilennummern zwischen den Zeilennummern des Programms** – die nächsthöhere Zeile wird angezeigt.

- **Zeilennummer 0 oder größer 32767**
  die Information "BAD LINE NUMBER" (falsche Zeilennummer) erscheint.
- **Nicht ganzzahlige Zeilennummer**
  Die Nachricht "INCORRECT STATEMENT" (Statement nicht korrekt) wird angezeigt.
- **Zeilennummer ohne Programm im Speicher**
  die Nachricht "CAN'T DO THAT" (Durchführung nicht möglich) wird angezeigt.
- **Unterbrechung der Auflistung**
  Sie können die Auflistung mit der Break-Taste **[FCTN 4]** (CLEAR) unterbrechen
- **Auflistung der Peripheriegeräte**
  LIST kann auch für die direkte Ausgabe an ein Peripheriegerät angewendet werden; z. B. LIST "TP". Damit wird das Programm ausgedruckt, wenn ein TI-Thermodrucker angeschlossen ist.
  Beachten Sie, daß der Name des Geräts in Anführungszeichen gesetzt werden muß. Weitere Informationen entnehmen Sie bitte den jeweiligen Anleitungen der Peripheriegeräte.

Zusammenfassung

Befehl	angezeigte Zeilen
LIST	alle Programmzeilen
LIST X	Programmzeile X
LIST X–Y	Programmzeile zwischen x und y inkl.
LIST X–	Programmzeilen ≧ X
LIST –Y	Programmzeilen ≦ Y

Beispiele:

```
>LIST 110
 110 B=-456.8
>LIST -110
 100 A=279.3
 110 B=-456.8
>LIST 120-
 120 PRINT A;B
 130 END
>LIST 90-120
 100 A=279.3
 110 B=-456.8
 120 PRINT A;B
```

```
>LIST 150-
 130 END
>LIST -90
 100 A=279.3
>LIST 105
 110 B=-456.8
>LIST 0
 * BAD LINE NUMBER
>LIST 33961
 * BAD LINE NUMBER
>LIST 32.7
 * INCORRECT STATEMENT
>NEW
>LIST
 * CAN'T DO THAT
```

RUN

Run *[Zeilennummer]*
Die Eingabe des RUN-Befehls bewirkt, daß der Ablauf des gespeicherten Programms beginnt. Vor dem Programmablauf werden die Werte aller numerischen Variablen auf Null, und die Werte aller String-Variablen auf null Zeichen (ein String mit 0 Zeichen) gesetzt. Alle Leerstellen, die zuvor für spezielle graphische Zeichen belegt waren, werden freigegeben

Wenn bei Eingabe des RUN-Befehls keine Zeilennummer angegeben wird, beginnt der Ablauf bei der Programmzeile mit der niedrigsten Nummer.

Wird bei Eingabe des RUN-Befehls die Zeilennummer angegeben, beginnt der Programmablauf bei dieser speziellen Programmzeile.
Beachten Sie im Beispiel rechts, daß der Wert von A Null bleibt, da der Programmablauf bei Zeile 110 beginnt.

Wenn Sie eine Zeilennummer angeben, die nicht im Programm enthalten ist, wird die Nachricht "BAD LINE NUMBER" angezeigt.

Bei Eingabe eines RUN-Befehls, wenn kein Programm gespeichert ist, wird die Nachricht "CAN'T DO THAT" angezeigt.

In Extended Basic kann der RUN-Befehl zusätzlich als Statement genutzt werden.*

* separat als Zubehör

Beispiele:

```
>NEW

>100 A=-16
>110 B=25
>120 PRINT A;B
>130 END
>RUN
 -16  25

 ** DONE **
```

```
>RUN 110
  0  25

 ** DONE **
```

```
>RUN 115

 * BAD LINE NUMBER
```

```
>NEW

>RUN

 * CAN'T DO THAT
```

BYE

BYE
Wenn Sie Ihre Arbeit abgeschlossen haben, und den BASIC-Modus abschalten wollen, geben Sie einfach den BYE-Befehl ein. Wir empfehlen, grundsätzlich den BYE-Befehl (anstatt QUIT) anzuwenden, wenn Sie den BASIC-Modus aufheben wollen. Bei Eingabe eines BYE-Befehls ist die erste Funktion, die der Computer ausführt, der Abschluß aller offenen Daten (siehe OPEN-Statement. Dann werden das gespeicherte Programm und alle Variablenwerte gelöscht. Schließlich wird der Computer so rückgesetzt, daß er jederzeit wieder in den BASIC-Modus geschaltet werden kann, wenn Sie dies wünschen. Nach Durchführung des BYE-Befehls erscheint wieder das Standardbild auf dem Schirm.

Beispiele:

```
>NEW

>100 LET X$="HELLO, GENIUS!"
>110 PRINT X$
>120 END
>RUN
 HELLO, GENIUS!

 ** DONE **

>BYE

--das Standardbild erscheint
  auf dem Bildschirm
```

NUMBER

{NUMBER / NUM} *[Anfangszeile] [,Inkrement]*

Wenn der NUMBER-Befehl eingegeben wird, erzeugt Ihr Computer automatisch die Zeilennummern. Ihr Computer befindet sich im sogenannten NUMBER MODE.

Die erste angezeigte Zahl nach Eingabe des NUMBER-Befehls ist die spezifizierte Anfangszeile.

Die 2. Zahl nach Eingabe des Kommas definiert das Inkrement (Zuwachszahl), mit deren Hilfe die nachfolgenden Zeilennummern gebildet werden.

Um die automatische Bildung von Zeilennummern zu beenden und den NUMBER MODE zu verlassen, drücken Sie unmittelbar nach Anzeige der erzeugten Zeilennummer die Taste ENTER. Die leere Zeile wird dann dem Programm nicht mehr beigefügt.

Beispiele:

```
>NEW

>NUMBER 10,5

>10 C=38.2
>15 D=16.7
>20 PRINT C;D
>25 END
>30 ENTER
>LIST
 10 C=38.2
 15 D=16.7
 20 PRINT C;B
 25 END
```

Werden weder Anfangszeile noch Inkrement spezifiziert, verwendet der Computer automatisch 100 als Anfangszeile und 10 als Inkrement.

```
>NEW

>NUM
>100 B$="HELLO!"
>110 PRINT B$
>120 END
>130 ENTER
```

Wird nur die Anfangszeile angegeben, verwendet der Computer 10 als Inkrement.

```
>NEW

>NUMBER 50
>50 C$="HI!"
>60 PRINT C$
>70 END
>80 ENTER
```

Wenn man nur ein Inkrement angibt, verwendet der Computer 100 als Nummer für die Anfangszeile. Beachten Sie im rechten Beispiel das Komma vor der Zahl 5.

```
>NEW

>NUM ,5
>100 Z=99.7
>105 PRINT Z
>110 END
>115 ENTER
```

NUMBER

Wenn Sie im NUMBER MODE arbeiten, und eine erzeugte Zeilennummer ist bereits belegt, dann wird die bestehende Programmzeile zusammen mit der Zeilennummer angezeigt. Beachten Sie, daß bei Anzeige einer bereits bestehenden Programmzeile im Nummern-Modus das Dialogzeichen (>) nicht ausgewiesen wird. Sie haben jetzt die Möglichkeit, die Zeile zu editieren. Informationen darüber finden Sie nachfolgend. Wenn Sie die Zeile nicht ändern wollen, drücken Sie bei Anzeige der Zeile einfach **[ENTER]**. Dann wird die nächste Zeilennummer gebildet. Wenn Sie im Nummernmodus eine Programmzeile eingeben und einen Fehler verursachen, wird die entsprechende Fehlernachricht wie gewöhnlich angezeigt, und dann wird die gleiche Zeilennummer erneut ausgewiesen. Tippen Sie die Zeile noch einmal richtig und geben Sie sie erneut ein. Wird im Nummernmodus eine Zeilennummer größer als 32767 gebildet, hebt der Computer den Nummernmodus auf.

Editieren im Nummern-Modus

Gleich, ob Sie im Nummern-Modus neue Zeilen eingeben oder bereits bestehende Programmzeilen ändern, Sie können alle speziellen Tasten zum Editieren anwenden. Einige der Tasten funktionieren im Nummern-Modus anders als im Befehlsmodus.

[ENTER] – Die verschiedenen Funktionen der Taste **[ENTER]** sind von der jeweiligen Situation abhängig:

- Wenn Sie **[ENTER]** unmittelbar nach Bildung einer Zeilennummer drücken, hebt der Computer den Nummern-Modus auf.
- Wenn Sie nach Bildung der Zeilennummer ein Statement eintippen, und dann **[ENTER]** drücken, wird die neue Zeile dem Programm beigefügt, und dann die nächste Zeilennummer gebildet.
- Wird eine bereits bestehende Programmzeile angezeigt, und Sie drükken unmittelbar nach der Anzeige **[ENTER]**, bleibt die Zeile unverändert im Programm. Dann wird die nächste Zeilennummer gebildet.
- Wird eine bereits bestehende Programmzeile angezeigt, und Sie löschen den gesamten Zeilentext (nur die Zeilennummer bleibt auf dem Bildschirm), hebt der Computer beim Drücken der Taste **[ENTER]** den Nummern-Modus auf. Die Programmzeile wird nicht aus dem Programm entfernt.
- Wenn Sie eine Zeile editieren, nachdem sie als bereits bestehende Programmzeile angezeigt ist und nach der Zeilennummer noch ein Text bleibt, wird beim Drücken der Taste **[ENTER]** diese Zeile durch die editierte (modifizierte) Zeile ersetzt. Dann wird die nächste Zeilennummer gebildet.

[FCTN 4] (CLEAR) – Immer, wenn Sie im Nummern-Modus die Clear-Taste drücken, rückt die momentane Zeile im Bildschirm nach oben, und der Computer hebt den Nummern-Modus auf. Alle Veränderungen, die vor Drücken der Clear-Taste an der Zeile vorgenommen wurden, werden ignoriert. Wenn Sie also gerade eine bestehende Programmzeile editiert haben, verändert sich die Programmzeile nicht. Wenn Sie gerade eine Zeile eingetippt haben, wird die Zeile nicht ins Programm aufgenommen.

Beispiele:

```
>NEW

>100 A=37.1
>110 B=49.6
>NUMBER 110
 110 B=49.6 ENTER
>120 PRINT A;B
>130 END
>140 ENTER
>LIST
 100 A=37.1
 110 B=49.6
 120 PRINT A;B
 130 END
```

RESEQUENCE

{RESEQUENCE / RES} *[Anfangszeile] [,Inkrement]*

Wollen Sie die eingegebene Zeilennummer ändern, können Sie dies mit dem RESEQUENCE-Befehl. Entsprechend der spezifizierten Anfangszeile und den Inkrement, erhalten alle Zeilen im Programm neue Zeilen-Nummern. Der RES-Befehl wird genauso wie der NUMBER-Befehl programmiert.

Anmerkung

Wird in einer Programmzeile eine Zeilennummer angegeben, die im vorliegenden Programm nicht existiert, dann wird diese Bezugnahme bei der Neuordnung der Zeilennummern auf 32767 geändert. Der Computer gibt keine Warnung oder Fehlermeldung.

Beispiele:

```
>NEW

>100 A=27.9
>110 B=34.1
>120 PRINT A;B
>130 END

>RESEQUENCE 20,5
>LIST
 20 A=27.9
 25 B=34.1
 30 PRINT A;B
 35 END
```

```
>NEW

>100 Z=Z+2
>110 PRINT Z
>120 IF Z=50 THEN 150
>130 GO TO 100
>140 END
>RES 10,5
>LIST
 10 Z=Z+2
 15 PRINT Z
 20 IF Z=50 THEN 32767
 25 GO TO 10
 30 END
```

BREAK

BREAK *Liste der Zeilennummern*

Bei Eingabe des BREAK-Befehls werden an den angegebenen Zeilennummern Stopp-Stellen vorgesehen. Diese Stopp-Stellen (Programmunterbrechungen) dienen im allgemeinen zur Erleichterung der Fehlersuche. Wenn Sie an einer bestimmten Zeile mit dem BREAK-Befehl eine Programmunterbrechung veranlassen, weisen Sie damit den Computer an, den Programmlauf vor der Durchführung des Statements in dieser Zeile zu stoppen.

Die Liste der Zeilennummern teilt dem Computer mit, wo Sie Stopp-Stellen setzen wollen. Die Zeilennummern werden durch Kommas getrennt (zum Beispiel: BREAK 10, 23, 35). Natürlich kann die Liste auch aus nur einer Zeilennummer bestehen.

Wenn beim Programmablauf eine Zeile erreicht wird, wo eine Programmunterbrechung vorgesehen ist, wird der Ablauf vor der Durchführung des Statements dieser Zeile gestoppt. Bei Unterbrechung des Programmablaufs durch eine Stoppstelle wird die Nachricht "BREAK-POINT AT ZEILENNUMMER" ("Stopp-Stelle bei Zeilennummer") angezeigt, und der Computer weist Sie mit dem blinkenden Positionszeiger an, einen Befehl einzugeben.

```
>NEW

>100 A=26.7
>110 C=19.3
>120 PRINT A
>130 PRINT C
>140 END


>BREAK 110



>RUN

 * BREAKPOINT AT 110

>□
```

BREAK

Wenn das Programm den Ablauf wegen einer Stopp-Stelle unterbricht, können Sie jedes Command oder jedes Statement, das als Command verwendet werden kann, eingeben.
Es gibt keine Veränderungen bei den variablen Werten, wenn Sie nicht ein Statement eingeben, das einen neuen Wert zuordnet.
Beachten Sie, daß im Beispiel rechts C immer noch gleich Null ist, weil die Zuordnung im Statement 110 nicht durchgeführt wurde.

Der BREAK-Befehl kann auch als Programmbefehl verwendet werden. Wird der BREAK-Befehl als Statement zusammen mit einer Zeilenliste eingegeben, werden die Stoppstellen bei den spezifizierten Zeilennummern programmiert. Die Löschung der in dieser Form programmierten Stoppstellen erfolgt wie oben beschrieben. Bedenken Sie aber, daß bei Eingabe des BREAK-Befehls als Statements mit einer Zeilenliste die Stoppstellen bei jeder Durchführung des Statements neu gesetzt werden.

Wird der BREAK-Befehl als Statement ohne Zeilenliste eingegeben, wirkt das Statement selbst wie eine Stoppstelle. Bei jedem Durchlauf des Programms wird an dieser Stelle der Programmablauf unterbrochen. Die einzige Möglichkeit, die Programmunterbrechung beim BREAK-Statement zu verhindern, ist die Löschung der entsprechenden Zeile aus dem Programm.

Wenn Sie in der Zeilenliste eine Nummer angeben, die nicht im Programm vorkommt, wird die Warnung "BAD LINE NUMBER" angezeigt. Stoppstellen werden nur bei den Zeilennummern gesetzt, die tatsächlich Programmzeilen sind.

Beispiele:

```
>LIST 110
 110 C=19.3
>PRINT A;C
  26.7  0

>A=5.8

>PRINT A
  5.8
```

```
>NEW

>100 B=29.7
>110 BREAK 120,140
>120 H=15.8
>130 PRINT B
>140 PRINT H
>150 END
>RUN

 * BREAKPOINT AT 120
```

```
>110 BREAK
>RUN

 * BREAKPOINT AT 110
```

```
>110 BREAK 125,140
>RUN

 * WARNING:
   BAD LINE NUMBER IN 110
```

CONTINUE

{CONTINUE / CON}

Die Eingabe des CONTINUE-Befehls ist immer dann möglich, wenn der Programmablauf wegen einer Stoppstelle unterbrochen wurde. Beachten Sie, daß eine Programm-Unterbrechung auch dann erfolgt, wenn Sie die Break-Taste **[FCTN 4]** drücken, während das Programm läuft.

Eine Programmunterbrechung darf nicht durch den CONTINUE-Befehl aufgehoben werden, wenn Sie das Programm editiert haben (Einfügen, Löschen oder Änderung von Programmzeilen). Damit sollen Fehler vermieden werden, die daher rühren, daß ein überarbeitetes Programm irgendwo in der Mitte gestartet wird. Wenn Sie nach dem Editieren des Programms den CONTINUE-Befehl eingeben, wird die Nachricht "CAN'T CONTINUE" (Fortsetzung nicht möglich) auf dem Bildschirm angezeigt.

Beispiele:

```
>NEW

>100 A=9.6
>110 PRINT A
>120 END
>BREAK 110

>RUN

 * BREAKPOINT AT 110

>CONTINUE
  9.6

 ** DONE **

>BREAK 110

>RUN

 * BREAKPOINT AT 110

>100 A=10.1
>CONTINUE
 * CAN'T CONTINUE
```

UNBREAK

UNBREAK *[Liste der Zeilennummern]*

Man wendet den UNBREAK-Befehl an, um die Stoppstellen aus den in der Zeilenliste angegebenen Programmzeilen zu löschen.

Die Zeilenliste ist eine Auflistung der Zeilennummern, an denen Sie die Stoppstellen aufheben wollen. Die Nummern der Programmzeilen werden durch Kommas getrennt (zum Beispiel: UNBREAK 10, 23). Wenn Sie nur eine Zeilennummer in Ihrer Liste angeben, erübrigt sich das Komma.

```
>NEW

>100 A=26.7
>110 C=19.3
>120 PRINT A
>130 PRINT C
>140 END
>BREAK 110,130

>RUN

 * BREAKPOINT AT 110

>UNBREAK 130

>CONTINUE
  26.7
  19.3

 ** DONE **
```

UNBREAK

Bei Eingabe des UNBREAK-Befehls ohne Zeilenliste werden alle Stoppstellen aufgehoben, die durch einen BREAK-Befehl oder durch ein BREAK-Statement gesetzt waren. Beachten Sie, daß der UNBREAK-Befehl keine Wirkung auf ein BREAK-Statement ohne Zeilenliste hat. Die einzige Möglichkeit, die Unterbrechung des Programmablaufs bei einem BREAK-Statement ohne Zeilenliste zu verhindern, ist die Löschung der entsprechenden Programmzeile.

Der UNBREAK-Befehl kann auch als Statement (Anweisung) in einem Programm verwendet werden. Das UNBREAK-Statement wird genauso wie der UNBREAK-Befehl durchgeführt. Beachten Sie im Beispiel, daß das UNBREAK-Statement die bei Zeile 130 gesetzte Stoppstelle aufhob.

Wenn Sie in der Zeilenliste eine Nummer angeben, die zwar im gültigen Bereich liegt, aber nicht im Programm vorkommt, wird die Warnung "BAD LINE NUMBER" angezeigt. Stoppstellen werden nur bei den Zeilennummern aufgehoben, die tatsächlich Programmzeilen sind.

Beispiele:

```
>125 BREAK
>BREAK 100,120,130

>RUN

 * BREAKPOINT AT 100
>UNBREAK

>CONTINUE
  26.7

 * BREAKPOINT AT 125

>CONTINUE
  19.3

 ** DONE **
```

```
>BREAK 130

>125 UNBREAK 130
>RUN
  26.7
  19.3

 ** DONE **
```

```
>BREAK 130

>UNBREAK 130,105

 * WARNING:
   BAD LINE NUMBER

>RUN
  26.7
  19.3

 ** DONE **
```

TRACE

TRACE

Mit dem TRACE-Befehl können Sie die Reihenfolge beobachten, in der Ihr Computer beim Ablauf des Programms die einzelnen Statements durchgeführt. Nach Eingabe des TRACE-Befehls wird die Nummer jeder Programmzeile vor der Durchführung des Statements angezeigt. Der TRACE-Befehl dient überwiegend als Hilfe bei der Fehlersuche in einem Programm (zum Beispiel zum Auffinden von unendlichen Schleifen).

Beispiele:

```
>NEW

>100 PRINT "HI"
>110 B=27.9
>120 PRINT :B
>130 END
>TRACE

>RUN
 <100>HI
 <110><120>
  27.9
 <130>
 ** DONE **
```

UNTRACE

UNTRACE

Der UNTRACE-Befehl hebt die Wirkung des TRACE-Befehls auf. Die Eingabe des UNTRACE-Befehls als Statement in ein Programm ist möglich.

Beispiele:

```
>NEW

>100 FOR I=1 TO 2
>110 PRINT I
>120 NEXT I
>130 END
>TRACE

>RUN
 <100><110> 1
 <120><110> 2
 <120><130>
 ** DONE **

>UNTRACE

>RUN
  1
  2

 ** DONE **
```

EDIT

EDIT *Zeilennummer*
Zeilennummer

Eine Änderung von bereits existierenden Programmzeilen erfolgt über den Edit-Modus. Der Edit-Modus wird mit dem EDIT-Befehl, gefolgt von einer Zeilennummer, aufgerufen. Wenn Sie eine Zeilennummer angeben, die nicht im Programm enthalten ist, wird die Nachricht "BAD LINE NUMBER" angezeigt.

Mit der Eingabe des Edit-Modus wird die geforderte Programmzeile ohne Dialogzeichen (>) auf dem Bildschirm ausgewiesen.
Sie können auf dieser Zeile beliebige Modifikationen vornehmen, mit Ausnahme an der Zeilennummer selbst.

Nachstehend erhalten Sie eine Erklärung über die speziellen Tasten zum Editieren (Modifizieren) und über deren Funktionen im Edit-Modus.

[ENTER] – Bei Drücken der Taste **[ENTER]** werden alle vorgenommenen Veränderungen in der Programmzeile gespeichert, und der Computer hebt den Edit-Modus auf. Wenn Sie den gesamten Text der Programmzeile gelöscht und dann **[ENTER]** gedrückt haben, wird die Programmzeile ebenfalls gelöscht.

[FCTN] [↑] – Bei Drücken der Taste mit dem nach oben gerichteten Pfeil werden alle Veränderungen an der Programmzeile eingegeben und sind dann gespeichert. Die Programmzeile mit der nächstniedrigen Nummer wird anschließend zum Editieren angezeigt. Wenn keine Programmzeile mehr mit niedriger Nummer existiert, verläßt der Computer den Edit-Modus.

[FCTN] [↓] – Bei Drücken der Taste mit dem nach unten gerichteten Pfeil werden alle Veränderungen an der Programmzeile eingegeben und sind dann gespeichert. Die Programmzeile mit der nächsthöheren Nummer wird anschließend zum Editieren angezeigt. Wenn keine Programmzeile mit einer höheren Nummer existiert, verläßt der Computer den Edit-Modus.

[FCTN 4] (CLEAR) – Bei Drücken der Clear-Taste im Edit-Modus wird die augenblickliche Zeile im Bildschirm nach oben verschoben und der Computer verläßt den Edit-Modus. Alle Änderungen, die an der Zeile vor Drücken der Clear-Taste vorgenommen wurden, werden ignoriert, d. h., die bestehende Programmzeile bleibt unverändert.

SAVE

SAVE *Dateiname*

Mit dem SAVE-Befehl können Sie Ihr im Computer gespeichertes Programm auf ein Peripheriegerät übertragen (kopieren). Eine kurze Erklärung zeigen wir Ihnen am Beispiel Kassettenrekorder als Speichergerät.

Durch Eingabe des Dateinamens CS1 oder CS2 nach dem Schlüsselwort SAVE treffen Sie die Wahl, welchen Kassettenrekorder der Computer verwenden wird. Nachdem Sie Ihren Rekorder an den Computer angeschlossen haben, tippen Sie den SAVE-Befehl ein, drücken **[ENTER]**, und der Computer beginnt auf dem Bildschirm die Instruktionen zum besseren Verständnis der SAVE-Vorgänge aufzuzeigen. Gehen Sie nach den Anleitungen auf dem Bildschirm vor.

Auf der rechten Seite befinden sich die SAVE-Instruktionen vom Computer. Im Beispiel wird CS1 verwendet, die Verfahren gelten jedoch ebenso für CS2.

Wenn Sie den SAVE-Befehl eingeben, weist Sie der Computer an, wie der Kassettenrekorder zu bedienen ist (siehe Spalte Beispiele rechts). Nach der Kopie des Programms fragt der Computer, ob Sie das Band kontrollieren wollen, um sicherzustellen, das Programm wurde korrekt aufgezeichnet. Wenn Sie **N** drücken, erscheint der blinkende Positionszeiger links im Bildschirm. Sie können dann jeden beliebigen BASIC-Befehl eintippen.

Beim Drücken von **Y** erscheinen Anleitungen zur Aktivierung des Rekorders auf dem Schirm.

Anmerkung: Die Antworten, die nur aus einem Buchstaben (Y, N, R, etc.) bestehen, welche Sie während der SAVE-Routine eingeben, müssen großgeschrieben werden. Drücken Sie die SHIFT-Taste und gleichzeitig die erforderliche Buchstabentaste.

Beispiele:

```
>SAVE CS1

 * REWIND CASSETTE TAPE     CS1
   THEN PRESS ENTER

 * PRESS CASSETTE RECORD    CS1
   THEN PRESS ENTER

 * RECORDING

 * PRESS CASSETTE STOP      CS1
   THEN PRESS ENTER

 * CHECK TAPE (Y OR N)? Y

 * REWIND CASSETTE TAPE     CS1
   THEN PRESS ENTER

 * PRESS CASSETTE PLAY      CS1
   THEN PRESS ENTER

 * CHECKING

 * DATA OK

 * PRESS CASSETTE STOP      CS1
   THEN PRESS ENTER
```

SAVE

Im Falle eines Fehlers haben Sie drei Möglichkeiten:

- Sie drücken **R**, um das Programm erneut aufzuzeichnen. Die oben angegebenen Instruktionen leiten Sie an.
- Sie drücken **C**, um das Kontrollverfahren zu wiederholen. An dieser Stelle können Sie die Lautstärke und/oder den Klangregler neu einstellen.
- Sie drücken **E**, um das Aufzeichnungsverfahren zu beenden. Der Computer weist Sie an, die Kassette zu stoppen und **[ENTER]** zu drücken. Auf dem Bildschirm wird eine Fehlermeldung angezeigt. Dies bedeutet,daß die SAVE-Routine Ihr Programm nicht richtig aufzeichnete. Nach Überprüfung des Rekorders können Sie die Aufzeichnung des Programms erneut versuchen. Wenn der blinkende Positionszeiger wieder auf dem Schirm erscheint, geben Sie einen beliebigen BASIC-Befehl ein.

Wenn der SAVE-Befehl durchgeführt wird, bleibt das Programm im Computer-Speicher, gleichgültig, ob bei der Aufzeichnung ein Fehler auftrat oder nicht.

(Genauere Informationen finden Sie in dieser Anleitung unter „Übertragung/Eingabe von Daten".) Bedienungshinweise für das Disketten-System sind der Anleitung des Disketten-Steuergerätes zu entnehmen.

In Extended Basic kann man zusätzlich mit dem MERGE-Befehl zweigespeicherte Programme verknüpfen.*

Beispiele:

```
* ERROR - NO DATA FOUND
  PRESS R TO RECORD
  PRESS C TO CHECK
  PRESS E TO EXIT

          or

* ERROR IN DATA DETECTED
  PRESS R TO RECORD
  PRESS C TO CHECK
  PRESS E TO EXIT

* I/O ERROR 66
```

* separat als Zubehör

OLD

OLD *Dateiname*

Der OLD-Befehl kopiert ein zuvor mit SAVE aufgezeichnetes Programm in den Speicher des Computers. Anschließend kann man das Programm ablaufen lassen, auflisten oder verändern. Eine Erklärung über die Anwendung des Kassettenrekorders (CS1) in Verbindung mit dem OLD-Befehl erhalten Sie in diesem Abschnitt. Bedienungshinweise für das Disketten-System sind der Anleitung des Disketten-Steuergeräts zu entnehmen.

Nach Eingabe des OLD-Befehls und Drücken der Taste **[ENTER]** beginnt der Computer mit dem Aufzeigen von Instruktionen auf dem Bildschirm, um Sie bei den Verfahren anzuleiten.
Befolgen Sie die Bildschirmanweisungen.

Achten Sie darauf, daß der Rekorder angeschlossen und die richtige Kassette eingelegt ist.

Rechts in der Spalte Beispiele sehen Sie die Bildschirm-Instruktionen, die bei Eingabe des OLD-Befehls angezeigt werden.
Wenn der Computer das Programm nicht korrekt in den Speicher einlesen konnte, liegt ein Fehler vor und Ihnen steht eine der folgenden Alternativen offen:

- Sie drücken **R**, um das Einleseverfahren zu wiederholen. Kontrollieren Sie vorher, ob einer der Störfaktoren vorliegt, die in dem Kapitel zum Kassettenrekorder aufgelistet sind.
- Sie drücken **E**, um den Einlesevorgang abzuschließen. In der Anzeige erscheint eine Fehlermeldung mit dem Hinweis, daß das Programm nicht richtig in den Speicher des Computers eingelesen wurde.

Anmerkung: Die Antworten, die aus nur einem Buchstaben (**E** oder **R**) bestehen, welche Sie während der OLD-Routine eintippen, müssen großgeschrieben werden. Drücken Sie die SHIFT-Taste und gleichzeitig die erforderliche Buchstabentaste.
Wenn der blinkende Positionszeiger wieder auf den Bildschirm kommt, können Sie einen beliebigen BASIC-Befehl eingeben.
Auch wenn das Programm nicht korrekt in den Speicher des Computers eingelesen wurde, kann ein zuvor gespeichertes Programm oder ein Teil davon überschrieben werden. Ehe Sie fortfahren, empfiehlt sich daher eine Überprüfung des Speicherinhalts mit Hilfe des LIST-Befehls.

Beispiele:

```
>OLD CS1

 * REWIND CASSETTE TAPE    CS1
   THEN PRESS ENTER

 * PRESS CASSETTE PLAY     CS1
   THEN PRESS ENTER

 * READING

 * DATA OK

 * PRESS CASSETTE STOP     CS1
   THEN PRESS ENTER

            or

 * ERROR - NO DATA FOUND
   PRESS R TO READ
   PRESS E TO EXIT

 * I/O ERROR 56
```

```
>OLD CS1
 -- Das Programm wird
    eingelesen.
>LIST "TP"
 -- Das Programm wird auf dem
    Thermodrucker aufgelistet.
    Eine oder mehrere Zeilen
    können geändert werden.
```

```
>OLD CS1
 -- Das Programm wird
    eingelesen.
>RUN
 -- Das Programm läuft.
>LIST "TP"
 -- Das Programm wird auf dem
    Thermodrucker aufgelistet.
```

DELETE

DELETE { *Dateiname* / *Programm-Name* }

Der DELETE-Befehl gibt Ihnen die Möglichkeit, ein Programm oder eine Datei aus dem Dateisystem des Computers zu entfernen. Der Dateiname und der Programm-Name sind String-Ausdrücke. Wenn eine String-Konstante verwendet wird, muß sie in Anführungszeichen gesetzt werden.

Nachdem dieser Befehl hauptsächlich in Verbindung mit Diskettensystemen gebraucht wird, entnehmen Sie bitte die weiteren Informationen aus der Bedienungsanleitung des Disketten-Steuergerätes.

Beispiele:

```
>SAVE NAME$

>DELETE NAME$

>DELETE "CS1"
```

Allgemeine Programmbefehle (STATEMENTS)

Einführung

Dieser Abschnitt beschreibt die allgemeinen Programmbefehle, die keiner Ein-/Ausgabefunktion dienen. Zu ihnen gehören das LET-Statement, mit dem man den Variablen Werte zuordnen kann, sowie STOP, END, REMark und diejenigen Statements, die den Weg des Computers beim Ablauf Ihres Programms steuern. Diese Anweisungen zur Programmsteuerung (GO TO, ON-GO TO, IF-THEN-ELSE, For-TO-STEP und NEXT) erlauben die problemlose Programmierung von Schleifen (Loops) und bedingten und unbedingten Verzweigungen. In TI-BASIC darf nur ein Statement pro Zeile benutzt werden.

Extended Basic erlaubt mehrmals ein Statement pro Zeile, dadurch wird die Abarbeitungszeit eines Programms durch den Computer verkürzt, Speicherkapazität eingespart und dem Anwender die Möglichkeit gegeben, logische Einheiten (z. B. FOR-NEXT-Schleifen) in einer Zeile zu programmieren.*

* separates Zubehör

LET

[LET] *Variable = Ausdruck*

Mit dem LET-Statement (LET-Anweisung) können Sie den Variablen in Ihrem Programm Werte zuordnen. Der Computer berechnet den Ausdruck rechts vom Gleichheitszeichen und setzt dessen Wert in die links vom Gleichheitszeichen angegebene Variable ein.

Beispiele:

```
>NEW

>100 LET M=1000
>110 LET C=186000
>120 E=M*C^2
>130 PRINT E
>140 END
>RUN
  3.4596E+13

 ** DONE **
```

LET (Zuordnungs-Statement)

Variable und Ausdruck müssen in ihrer Art übereinstimmen: numerische Ausdrücke müssen numerischen Variablen zugeordnet sein String-Ausdrücke müssen String-Variablen zugeordnet sein. Die Regeln, die Überlauf und Bereichsunterschreitung bei der Berechnung von numerischen Ausdrücken steuern, gelten ebenfalls im LET-Statement. Eine eingehende Erklärung finden Sie unter „Numerischer Konstante". Wenn die Länge eines geprüften String-Ausdrucks 255 Zeichen überschreitet, wird der String rechts abgeschnitten und das Programm fortgesetzt. Eine Warnung erfolgt nicht.

Es ist möglich, in numerischen und in String-Ausdrücken Vergleichsoperatoren zu verwenden. Das Ergebnis einer Vergleichsoperation ist —1, wenn die Relation zutrifft, und 0, wenn die Relation nicht zutrifft.

Beispiele:

```
>NEW

>100 LET X$="HELLO, "
>110 NAME$="GENIUS!"
>120 PRINT X$;NAME$
>130 END
>RUN
 HELLO, GENIUS!

 ** DONE **
```

```
>NEW
>100 LET A=20
>110 B=10
>120 LET C=A>B
>130 PRINT A;B;C
>140 C=A<B
>150 PRINT A;B;C
>160 END
>RUN
  20  10 -1
  20  10  0

 ** DONE **
```

REMark

REM *Anmerkungen*

Mit dem REM-Statement können Sie Ihr Programm durch Einfügen von eigenen Kommentaren näher erläutern und dokumentieren. Der Computer beachtet die REM-Statements bei der Abarbeitung des Programmes nicht. Sie dienen nur zur Dokumentation oder Erläuterung für den Benutzer.

Für das REM-Statement kann jedes druckbare Zeichen verwendet werden. Die Länge des REM-Statements ist durch die Länge einer Programmzeile (112 Zeichen oder vier Bildschirmzeilen) limitiert.

```
>NEW

>100 REM COUNTING FROM 1 TO
 10
>110 FOR X=1 TO 10
>120 PRINT X;
>130 NEXT X
>140 END
>RUN
  1  2  3  4  5  6  7  8  9
  10
 ** DONE **
```

```
>NEW

>100 A=762
>110 B=425
>120 REM NOW PRINT THE SUM OF
 A AND B
>130 PRINT A+B
>140 END
>RUN
  1187

 ** DONE **
```

END

END

Mit dem END-Statement wird das Programm bei seiner Ausführung abgeschlossen. Es kann in TI-BASIC gleichbedeutend mit dem STOP-Statement angewendet werden. Obwohl das END-Statement an beliebiger Stelle im Programm erscheinen kann, wird es im allgemeinen in die letzte Zeilennummer des Programms gesetzt, und schließt somit das Programm physikalisch und logisch ab. Zwar können überall im Programm END-Statements enthalten sein, es ist aber üblich, das STOP-Statement zu verwenden, wenn man im Programm selbst bestimmte Grenzpunkte setzen will. In TI-BASIC ist ein END-Statement im Programm nicht erforderlich.

Beispiele:

```
>NEW

>100 A=10
>110 B=20
>120 C=A*B
>130 PRINT C
>140 END
>RUN
  200

 ** DONE **
```

STOP

STOP

Das STOP-Statement schließt das Programm bei seiner Ausführung ab. Es kann in TI-BASIC im Wechsel mit dem END-Statement angewendet werden. STOP-Statements dürfen an beliebiger Stelle im Programm enthalten sein. Mehrere STOP-Statements in einem Programm sind möglich.

```
>NEW

>100 CALL CLEAR
>110 FOR I=1 TO 15
>120 CALL HCHAR(1,1,42,768)
>130 GOSUB 160
>140 NEXT I
>150 STOP
>160 F=I
>170 B=I+1
>180 CALL COLOR(2,F,B)
>190 RETURN
>200 END
>RUN

  --BILDSCHIRM FÜLLT SICH
    MIT STERNCHEN UND WECHSELT
    15MAL DIE FARBE.

 ** DONE **
```

GOTO

GOTO / **GO TO** *Zeilennummer*

Das GOTO-Statement erlaubt Verzweigungen innerhalb eines Programms nach vorn oder zurück.

Wenn der Computer ein GOTO-Statement erreicht, erfolgt ein Sprung zu der mit dem Statement angegebenen Zeilennummer. Diese Programmanweisung bezeichnet man als unbedingten Sprung oder unbedingte Verzweigung.

Im Programm rechts enthält die Zeile 150 einen unbedingten Sprung. An dieser Stelle springt der Computer immer zu Zeile 120.

Wenn Sie den Computer anweisen, zu einer Zeilennummer zu verzweigen, die nicht im Programm existiert, wird der Ablauf unterbrochen und die Nachricht "BAD LINE NUMBER" angezeigt.

Beachten Sie, daß die Leerstelle zwischen den Wörtern GO und TO Ihrer freien Wahl überlassen ist.

```
100 CALL CLEAR
110 REM  GOTO BEISPIEL
120 PRINT "TEXAS"
130 PRINT "INSTRUMENTS"
140 PRINT "  "
150 GOTO 120
160 END
```

ON-GOTO

ON *numerischer Ausdruck* { **GOTO** / **GO TO** } *Zeilennummer [,Zeilennummer]...*

Das ON-GOTO-Statement weist den Computer an, abhängig vom Wert des numerischen Ausdrucks zu einer von mehreren Programmzeilen zu springen.

Der Computer berechnet zuerst den numerischen Ausdruck und rundet das Ergebnis auf eine ganze Zahl. Diese ganze Zahl wird dann ein Zeiger für den Computer, der darauf hinweist, welche Programmzeile im ON-GOTO-Statement als nächste durchzuführen ist. Ist der Wert des numerischen Ausdrucks 1, verzweigt der Computer zu der ersten Zeilennummer, die im ON-GOTO-Statement angegeben ist. Ist der Wert 2, erfolgt der Sprung zu dem Statement mit der zweiten Zeilennummer etc.

Ist der gerundete Wert des numerischen Ausdrucks kleiner als 1 oder größer als die Zahl der im ON-GOTO-Statement aufgelisteten Zeilennummern, unterbricht das Programm den Ablauf und die Nachricht "BAD VALUE IN xx" (falscher Wert bei xx) wird angezeigt. Wenn die Zeilennummer-Angabe außerhalb des Bereichs der Zeilennummern in Ihrem Programm liegt, wird die Nachricht "BAD LINE NUMBER" angezeigt und das Programm unterbrochen.

Beispiele:

```
>NEW

>100 REM HOW DOES ON-GOTO
 WORK?
>110 INPUT X
>120 ON X GOTO 130,150,170,19
 0,210
>130 PRINT "X=1"
>140 GOTO 110
>150 PRINT "X=2"
>160 GOTO 110
>170 PRINT "X=3"
>180 GOTO 110
>190 PRINT "X=4"
>200 GOTO 110
>210 END
>RUN
 ? 2
 X=2
 ? 1.2
 X=1
 ? 3.7
 X=4
 ? 6

 * BAD VALUE IN 120
```

IF-THEN-ELSE

IF { *Vergleich* / *numerischer Ausdruck* } **THEN** *Zeile-1* [**ELSE** *Zeile-2*]

Das IF-THEN-ELSE-Statement erlaubt die Änderung der normalen Folge im Programmablauf durch Anwendung einer bedingten Verzweigung.

Der Computer ermittelt den Ausdruck, den Sie in das Statement einbezogen haben, zum Beispiel A > 50. Wenn diese Relation zutrifft, springt der Computer zu Zeile-1, die dem Wort THEN folgt. Ist die Bedingung nicht erfüllt, führt der Computer die Verzweigung zur Zeile-2 durch, die nach dem Wort ELSE angegeben ist. Wenn ELSE ausgelassen ist, setzt der Computer das Programm mit der nächsten Programmzeile fort.

Die zulässige Vergleichsoperatoren im TI-BASIC sind:

- ist gleich (=)
- ungleich (< >)
- kleiner als (<)
- kleiner oder gleich (< =)
- größer als (>)
- größer oder gleich (> =)

Nachstehend einige gültige Vergleiche:

- A > 7
- A$ < "YES"
- CHR$ (L) = "A"
- (A + B)/2 < >AVG
- (A$&C$)> =D$

```
>NEW

>100 REM FIND THE LARGEST OF
 A SET OF NUMBERS
>110 INPUT "HOW MANY VALUES?"
 :N
>120 INPUT "VALUE?":A
>130 L=A
>140 N=N-1
>150 IF N<=0 THEN 180
>160 INPUT "VALUE?":A
>170 IF L>A THEN 140 ELSE 130

>180 PRINT L;"IS THE LARGEST"

>190 END
>RUN
 HOW MANY VALUES?3
 VALUE?456
 VALUE?321
 VALUE?292
  456 IS THE LARGEST

 ** DONE **
```

IF-THEN-ELSE

Ein numerischer Ausdruck muß mit einem anderen numerischen Ausdruck und ein String-Ausdruck mit einem anderen String-Ausdruck verglichen werden. Der Vergleich von numerischen Ausdrücken erfolgt algebraisch. String-Ausdrücke werden von links nach rechts, Zeichen für Zeichen verglichen.
Zu diesem Zweck werden die ASCII-Zeichenkodes verwendet.
Ein Zeichen mit einem niedrigeren ASCII-Kode wird kleiner erachtet als ein Zeichen mit höherem ASCII-Kode. Auf diese Weise können Sie Strings in einer numerischen und alphabetischen Ordnung sortieren. Ist ein String länger als der andere, wird der Vergleich für jedes Zeichen im kürzeren String durchgeführt. Wenn dabei kein Unterschied festgestellt wird, betrachtet der Computer den längeren String automatisch als den größeren.

Ein alternatives Format des IF-THEN-ELSE-Statements ist die Verwendung eines numerischen Ausdrucks ohne Angabe einer Beziehung.

Im Beispiel rechts ermittelt der Computer den Ausdruck A + B. Ist das Resultat 0, wird der Ausdruck als falsch behandelt. Ein Ergebnis ungleich null wird als richtig angesehen.

Dies ist dasselbe wie:

IF *Ausdruck* < > 0 THEN *Zeile-1*

Beispiele:

```
>NEW

>100 INPUT "A$ IS ":A$
>110 INPUT "B$ IS ":B$
>120 IF A$=B$ THEN 160
>130 IF A$<B$ THEN 180
>140 PRINT "B$ IS LESS"
>150 GOTO 190
>160 PRINT "A$=B$"
>170 GOTO 190
>180 PRINT "B$ IS GREATER"
>190 END
>RUN
 A$ IS TEXAS
 B$ IS TEX
 B$ IS LESS

 ** DONE **

>RUN
 A$ IS TAXES
 B$ IS TEX
 B$ IS GREATER

 ** DONE **
```

```
>NEW

>100 INPUT "A IS ":A
>110 INPUT "B IS ":B
>120 IF A+B THEN 150
>130 PRINT "RESULT IS ZERO,EX
 PRESSION FALSE"
>140 GOTO 100
>150 PRINT "RESULT IS NON-ZER
 O,EXPRESSION TRUE"
>160 GO TO 100
>RUN
 A IS 2
 B IS 3
 RESULT IS NON-ZERO,EXPRESSIO
 N TRUE
 A IS 2
 B IS -2
 RESULT IS ZERO,EXPRESSION FA
 LSE
```

(Durch CLEAR wird die Schleife abgebrochen)

FOR-TO-STEP

FOR *Kontrollvariable* = *Anfangswert* **TO** *Grenze* [**STEP** *Inkrement*]

Das FOR-TO-STEP-Statement verwendet man zur Programmierung von sich wiederholenden (iterativen) Prozessen (Schleifen). Zusammen mit dem NEXT-Statement konstruiert man eine FOR-NEXT-Schleife. Wenn man STEP (Schrittgröße) ausläßt, verwendet der Computer ein Inkrement von +1.

Die Kontrollvariable ist eine numerische Variable, die als Zähler für die Schleife wirkt. Wenn man das FOR-TO-STEP-Statement ausführt, wird die Kontrollvariable auf einen Anfangswert gesetzt. Dann läßt der Computer die Programm-Statements ablaufen, bis ein NEXT-Befehl erreicht wird. Wenn das NEXT-Statement durchgeführt ist, erhöht der Computer die Kontrollvariable um die mit STEP spezifierte Schrittgröße. (Ist das Inkrement ein negativer Wert, wird die Kontrollvariable tatsächlich um die Schrittgröße reduziert.) Der Computer vergleicht dann die Kontrollvariable mit dem für Grenze angegebenen Wert. Wenn die Kontrollvariable die Grenze noch nicht überschreitet, wiederholt der Computer die Statements nach FOR-TO-STEP, bis wiederum das NEXT-Statement erreicht und durchgeführt wird. Ist der neue Wert für die Kontrollvariable größer als die Grenze (positives Inkrement) oder kleiner als die Grenze (negatives Inkrement), verläßt der Computer die Schleife und fährt mit dem Statement nach NEXT fort. Der Wert der Kontrollvariablen wird beim Verlassen der FOR-NEXT-Schleife nicht verändert.

Wie oft die FOR-NEXT-Schleife durchgeführt wird, kontrollieren Sie durch die Werte, die Sie dem FOR-TO-STEP-Statement zuordnen. Die Grenze, und wahlweise die Schrittgröße, sind numerische Ausdrücke, die einmal während eines Schleifendurchlaufs berechnet werden (bei Erreichen des FOR-TO-STEP-Statements), und die bis zur Beendigung der Schleife wirksam bleiben. Jede Veränderung dieser Werte während eines Schleifendurchlaufs hat keinen Einfluß auf die Anzahl der Schleifen. Ist der Wert des Inkrements 0, zeigt der Computer die Fehlernachricht "BAD VALUE IN xx" (falscher Wert in xx) an und unterbricht den Programmablauf.

Beispiele:

```
>NEW

>100 REM COMPUTING SIMPLE
 INTEREST FOR 10 YEARS
>110 INPUT "PRINCIPLE? ":P
>120 INPUT "RATE? ":R
>130 FOR YEARS=1 TO 10
>140 P=P+(P*R)
>150 NEXT YEARS
>160 P=INT(P*100+.5)/100
>170 PRINT P
>180 END
>RUN
 PRINCIPLE? 100
 RATE? .0775
  210.95

 ** DONE **
```

```
>NEW

>100 REM EXAMPLE OF
 FRACTIONAL INCREMENT
>110 FOR X=.1 TO 1 STEP .2
>120 PRINT X;
>130 NEXT X
>140 PRINT :X
>150 END
>RUN
  .1  .3  .5  .7  .9
  1.1

 ** DONE **
```

```
>NEW

>100 L=5
>110 FOR I=1 TO L
>120 L=20
>130 PRINT L;I
>140 NEXT I
>150 END
>RUN
  20  1
  20  2
  20  3
  20  4
  20  5

 ** DONE **
```

FOR-TO-STEP

Nach Eingabe des RUN-Befehls, aber vor der Durchführung des Programms kontrolliert der Computer, daß Sie die gleiche Anzahl von FOR-TO-STEP- und NEXT-Statements haben. Bei ungleicher Anzahl wird die Nachricht "FOR-NEXT-ERROR" (FOR-NEXT-Fehler) angezeigt und das Programm läuft nicht ab.

Beispiele:

Die Änderung des Wertes für die Kontroll-Variable während der Schleifen-Durchführung beeinflußt die Anzahl der Schleifen.

```
>NEW

>100 FOR I=1 TO 10
>110 I=I+1
>120 PRINT I
>130 NEXT I
>140 PRINT I
>150 END
>RUN
  2
  4
  6
  8
  10
  11

 ** DONE **
```

In TI-BASIC werden die Ausdrücke für den Anfangswert, die Grenze und das Inkrement berechnet, ehe der Anfangswert der Kontrollvariablen zugeordnet wird. Auf diese Weise wird im Programm rechts in Zeile 110 der Wert 5 der Grenze zugeordnet, ehe die Zuordnung eines Wertes für 1 als Kontrollvariable erfolgt. Die Schleife wird fünfmal wiederholt. nicht nur einmal.

```
>NEW

>100 I=5
>110 FOR I=1 TO I
>120 PRINT I;
>130 NEXT I
>140 END
>RUN
 1  2  3  4  5
 ** DONE **
```

Das Vorzeichen der Kontrollvariablen kann sich während der Durchführung einer FOR-NEXT-Schleife ändern.

```
>NEW

>100 FOR I=2 TO -3 STEP -1
>110 PRINT I;
>120 NEXT I
>130 END
>RUN
 2  1  0 -1 -2 -3
 ** DONE **
```

Bei der Ausführung des FOR-Statements prüft der Computer vor dem Schleifendurchlauf, ob die Grenze den Anfangswert im FOR-Statement erreicht. Dieser muß nicht 1 sein. Der Computer kann die Zählung mit einem beliebigen numerischen Wert beginnen. Ist jedoch der Anfangswert größer als die Grenze und das Inkrement positiv, wird die Schleife überhaupt nicht ausgeführt. Der Computer setzt in diesem Fall das Programm mit dem Statement nach der Schleife fort. Ebenso wird die Schleife auch dann nicht ausgeführt, wenn das Inkrement negativ ist, und Sie einen Anfangswert zuordnen, der niedriger ist als die Grenze.

```
>NEW

>100 REM INITIAL VALUE TOO
 GREAT
>110 FOR I=6 TO 5
>120 PRINT I
>130 NEXT I
>140 END
>RUN

 ** DONE **
```

FOR-TO-STEP

FOR-NEXT-Schleifen können ineinandergreifen, d.h., eine FOR-NEXT-Schleife kann völlig in einer anderen enthalten sein. Sie müssen jedoch sorgfältig auf die folgenden Regeln achten:

- Jedes FOR-TO-STEP-Statement muß mit einen NEXT-Statement gepaart sein.
- Für jede ineinandergeschachtelte FOR-NEXT-Schleife sind verschiedene Kontrollvariable zu verwenden.
- In einer FOR-NEXT-Schleife darf nicht nur ein Teil einer anderen FOR-NEXT-Schleife enthalten sein, sondern die gesamte zweite Schleife.

Andernfalls unterbricht der Computer den Ablauf des Programms und druckt die Fehlermeldung "CAN'T DO THAT IN xx" (Durchführung bei xx nicht möglich), wenn eine FOR-NEXT-Schleife eine andere überschneidet.

Mit den Statements GOTO und IF-THEN-ELSE können Sie aus einer FOR-NEXT-Schleife springen, es ist aber nicht möglich, mit diesen Statements in eine FOR-NEXT-Schleife hineinzuverzweigen. Zum Verlassen und zur Rückkehr zu einer FOR-NEXT-Schleife kann man GOSUB-Statements verwenden. Achten Sie darauf, daß Sie nicht die gleiche Kontrollvariable für FOR-NEXT-Schleifen verwenden, die Sie vielleicht in Ihren Unterprogrammen haben.

Beispiele:

```
>NEW

>100 REM FIND THE LOWEST
 THREE DIGIT NUMBER EQUAL TO
 THE SUM OF THE CUBES OF ITS
 DIGITS
>110 FOR HUND=1 TO 9
>120 FOR TENS=0 TO 9
>130 FOR UNITS=0 TO 9
>140 SUM=100*HUND+10*TENS+UNI
 TS
>150 IF SUM<>HUND^3+TENS^3+UN
 ITS^3 THEN 180
>160 PRINT SUM
>170 GOTO 210
>180 NEXT UNITS
>190 NEXT TENS
>200 NEXT HUND
>210 END
>RUN
  153

 ** DONE **
```

```
>NEW

>100 FOR I=1 TO 3
>110 PRINT I
>120 GOSUB 140
>130 NEXT I
>140 FOR I=1 TO 5
>150 PRINT I;
>160 NEXT I
>170 RETURN
>180 END
>RUN
  1
  1  2  3  4  5
 * CAN'T DO THAT IN 130
```

NEXT

NEXT *Kontrollvariable*

Das NEXT-Statement wird zur Bildung einer Schleife immer mit dem FOR-TO-STEP-Statement gepaart. Die Kontrollvariable ist mit derjenigen des entsprechenden FOR-TO-STEP-Statements identisch.

Tatsächlich steuert man mit dem NEXT-Statement, ob der Computer die Schleife wiederholt, oder mit der Programmzeile nach dem NEXT-Statement fortfährt.

Wenn der Computer auf ein NEXT-Statement trifft, addiert er das zuvor in der STEP-Eintragung ermittelte Inkrement zur Kontrollvariablen. Dann prüft er die Kontrollvariable, ob sie die zuvor ermittelte Grenze überschreitet, die im FOR-TO-STEP-Statement angegeben ist. Wenn die Kontrollvariable die Grenze nicht überschreitet, wird die Schleife wiederholt.

Beispiele:

```
>NEW

>100 REM COUNTING FROM 1 TO
 10
>110 FOR X=1 TO 10
>120 PRINT X;
>130 NEXT X
>140 END
>RUN
 1  2  3  4  5  6  7  8  9
 10
 ** DONE **
```

```
>NEW

>100 REM ROCKET COUNTDOWN
>110 CALL CLEAR
>120 FOR I=10 TO 1 STEP -1
>130 PRINT I
>140 FOR DELAY=1 TO 200
>150 NEXT DELAY
>160 CALL CLEAR
>170 NEXT I
>180 PRINT "BLAST OFF!"
>190 REM CHANGE SCREEN COLOR
>200 FOR COLOR=2 TO 16 STEP 2
>210 CALL SCREEN(COLOR)
>220 FOR DELAY=1 TO 100
>230 NEXT DELAY
>240 NEXT COLOR
>250 END
>RUN

--Computer zeigt Countdown an

 BLAST OFF!

--Bildschirm ändert 8mal
  die Farbe.

 ** DONE **
```

INPUT-OUTPUT-STATEMENTS (Ein-/Ausgabeanweisungen)

Einführung

INPUT-OUTPUT-Statements ermöglichen den Transfer von Daten in ein Programm und aus einem Programm. Der vorliegende Abschnitt beschreibt diese Statements (PRINT, DISPLAY, INPUT, READ, DATA, RESTORE).

Die Dateneingabe in das Programm geschieht auf drei verschiedene Arten:

- über die Tastatur - mit dem INPUT-Statement (INPUT - Eingabe)
- intern über das Programm selbst - mit den Statements READ, DATA und RESTORE
- von Dateien, die auf Peripheriegeräten gespeichert sind - wiederum mit dem INPUT-Statement.

Die Datenausgabe kann über zwei Arten von Ausgabegeräten erfolgen:

- über den Bildschirm mit den Statements PRINT oder DISPLAY
- als Dateien, die auf Peripheriegeräten gespeichert sind, mit dem PRINT-Statement.

In zwei weiteren Abschnitten dieser Anleitung werden zusätzliche Ein-/Ausgabemöglichkeiten (Ihres Computers TI-99/4A) beschrieben.

Der Abschnitt „Dateiverarbeitung" unterstützt Sie beim Erstellen von Statements, die man in Verbindung mit Peripheriegeräten anwendet. Da die Leistungsfähigkeit Ihres Video-Computers durch Grafik-, Farb- und Tonfunktionen gesteigert wurde, dienen viele festprogrammierte Unterprogramme ebenfalls der Ein- und Ausgabe. Informationen, wie diese Eigenschaften zu nutzen sind, finden Sie im Abschnitt „Farbgrafiken und akustische Signale".

INPUT

INPUT [*Eingabe-Dialog:*] *Variablenliste*

(Informationen über die Anwendung des INPUT-Statements in Verbindung mit einer Datei finden Sie im Abschnitt über die "Dateiverarbeitung".

Das INPUT-Statement wird zur Dateneingabe über die Tastatur verwendet und bewirkt eine Programmunterbrechung, bis gültige Daten von der Tastatur eingegeben sind. Obwohl der Computer im allgemeinen bis zu einer Programmzeile (4 Bildschirmzeilen) für jedes INPUT-Statement akzeptiert, kann es vorkommen, daß er eine lange Werteliste zurückweist. Wenn Sie nach Eintasten einer Eingabezeile die Nachricht "LINE TOO LONG" (zu lange Zeile) erhalten, müssen Sie das zu umfangreiche INPUT-Statement in mindestens zwei separate Statements aufteilen.

Eingabe des INPUT-Statements

Der Eingabe-Dialog (input-prompt) ist ein String-Ausdruck, der auf dem Bildschirm die Werte angibt, die Sie im Augenblick eingeben sollen. Das Einschließen des Eingabe-Dialogs in ein INPUT-Statement ist Ihnen selbst überlassen. Wenn der Computer ein INPUT-Statement ohne Eingabedialog durchführt, zeigt er ein Fragezeichen (?) an, gefolgt von einem Zwischenraum, und erwartet Ihre Dateneingabe.

Beispiele:

```
>NEW

>100 INPUT B
>110 PRINT B
>120 END
>RUN
 ? 25
  25

 ** DONE **
```

INPUT

Bei Anwendung eines Eingabe-Dialogs muß der String-Ausdruck mit einem Doppelpunkt abgeschlossen werden. Wenn der Computer diesen Typus des INPUT-Statements durchführt, zeigt er die Dialognachricht auf dem Bildschirm an und erwartet Ihre Dateneingabe.

Beispiele:

```
>NEW

>100 INPUT "COST OF CAR?":B
>110 A$="TAX?"
>120 INPUT A$:C
>130 INPUT "SALES "&A$:X
>140 PRINT B;C;X
>150 END
 RUN
 COST OF CAR?5500
 TAX?500
 SALES TAX?500
  5500  500  500

 ** DONE **
```

Die Variablenliste enthält diejenigen Variablen, denen bei Durchführung des INPUT-Statements Werte zugeordnet werden. Die Variablennamen in dieser Liste werden durch Kommata getrennt, und sie können numerische und/oder String-Variable sein.

```
>NEW

>100 INPUT A,B$,C,D
>110 PRINT A:B$:C:D
>120 END
 RUN
 ? 10,HELLO,25,3.2
  10
 HELLO
  25
  3.2

 ** DONE **
```

Reaktion auf das INPUT-Statement

Bei Durchführung eines INPUT-Statements müssen die den Variablen entsprechenden Werte in der gleichen Reihenfolge eingegeben werden, wie sie im Statement selbst aufgelistet sind.

Alle Werte müssen in einer Zeile (bis zu vier Bildschirmzeilen) eingegeben und durch Kommata voneinander abgegrenzt werden. Bei der Eingabe von String-Werten kann man den String in Anführungszeichen setzen. Die Anführungszeichen müssen gesetzt werden, wenn der String, den Sie eingeben wollen, ein Komma, ein einleitendes Anführungszeichen oder einleitende bzw. abschließende Leerstellen enthält.

```
>NEW

>100 INPUT A$
>110 PRINT A$::
>120 INPUT B$
>130 PRINT B$::
>140 INPUT C$
>150 PRINT C$::
>160 INPUT D$
>170 X=500
>180 PRINT D$;X::
>190 INPUT E$
>200 PRINT E$
>210 END
 RUN
 ? "JONES, MARY"
 JONES, MARY

 ? """HELLO THERE"""
 "HELLO THERE"

 ? "JAMES B. SMITH, JR."
 JAMES B. SMITH, JR.

 ? "SELLING PRICE IS "
 SELLING PRICE IS  500

 ? TEXAS
 TEXAS

 ** DONE **
```

INPUT

Den Variablen werden die Werte von links nach rechts in der Variablenliste zugeordnet. Auf diese Weise werden Indexausdrücke in der Variablenliste solange nicht berechnet, bis den Variablen zur Linken die Werte zugeordnet wurden.

In Extended Basic kann mit dem LINPUT-Statement die Eingabe ohne Editieren erfolgen. Das ACCEPT-Statement erlaubt die Eingabe von nahezu allen Bildschirm-Stellen aus.*

Die Eingabeinformationen werden vom Computer auf ihre Gültigkeit hin kontrolliert. Sind die Eingabedaten ungültig, erscheint die Nachricht "WARNING: INPUT ERROR. TRY AGAIN" (Warnung: Eingabefehler. Eingabe wiederholen) auf dem Bildschirm, und Sie müssen die Zeile neu eingeben. Nachstehend einige Ursachen für diese Fehlermeldung:

- Sie versuchen, Daten einzugeben, die mehr oder weniger Werte enthalten, als das INPUT-Statement fordert.
- Sie versuchen, eine String-Konstante einzugeben, wenn eine Zahl gefordert wird. (Bedenken Sie, daß umgekehrt eine Zahl ein gültiger String ist; Sie können also eine Zahl eingeben, wenn eine String-Konstante gefordert wird.)

Wenn Sie eine Zahl eingeben, die einen Überlauf verursacht, wird die Nachricht "WARNING: NUMBER TOO BIG. TRY AGAIN" (Warnung: zu große Zahl. Eingabe wiederholen) auf dem Bildschirm angezeigt, und Sie müssen die Zeile neu eingeben.

Bei Eingabe einer Zahl, die eine Bereichsunterschreitung bewirkt, wird der Wert ohne Warnung durch Null ersetzt.

In Extended Basic wird die Fehleranzeige durch den Computer erheblich ausgebaut. Feher können wesentlich schneller lokalisiert werden.*

* separat als Zubehör

Beispiele:

```
>NEW

>100 INPUT I,A(I)
>110 PRINT I:A(3)
>120 END
 RUN
 ? 3,7
  3
  7

 ** DONE **
```

```
>NEW

>100 INPUT A,B$
>110 PRINT A;B$
>120 END
>RUN
 ? 12,HI,3

 * WARNING:
   INPUT ERROR IN 100
 TRY AGAIN: HI,3

 * WARNING:
   INPUT ERROR IN 100
 TRY AGAIN: 23,HI
  23 HI

 ** DONE **
```

```
>NEW

>100 INPUT A
>110 PRINT A
>120 END
>RUN
 ? 23E139

 * WARNING:
   NUMBER TOO BIG IN 100
 TRY AGAIN: 23E-139
 0

 ** DONE **
```

READ

READ *Variablenliste*

Das READ-Statement erlaubt das Lesen von Daten, die in den DATA-Statements programm-intern gespeichert sind.

Die Variablenliste gibt diejenigen Variablen an, denen Werte zugeordnet werden sollen. Die Variablenbezeichnungen in dieser Liste werden durch Kommata abgegrenzt. Die Variablenliste kann aus numerischen und/oder String-Variablen bestehen.

Beispiele:

```
>NEW

>100 FOR I=1 TO 3
>110 READ X,Y
>120 PRINT X;Y
>130 NEXT I
>140 DATA 22,15,36,52,48,96.5

>150 END
>RUN
  22  15
  36  52
  48  96.5

 ** DONE **
```

DATA

DATA *Datenliste*

Das DATA-Statement erlaubt die programm-interne Speicherung von Daten. Die Daten in den Datenlisten werden über die READ-Statements wiedergewonnen, wenn das Programm abläuft. Die Datenliste enthält die Werte, die den in der Variablenliste eines READ-Statements angegebenen Variablen zuzuordnen sind. Die Elemente in der Datenliste werden durch Kommas begrenzt. Wenn ein Programm ein DATA-Statement erreicht, geht es ohne eine andere Wirkung zum nächsten Statement weiter.

DATA-Statements können an beliebiger Stelle in einem Programm erscheinen, aber ihre Reihenfolge ist wichtig. Die Daten werden der Reihe nach aus den Datenliste gelesen, beginnend mit dem ersten Element der ersten DATA-Anweisung. Wenn Ihr Programm mehr als ein DATA-Statement enthält, erfolgt der Lesevorgang nach Zeilennummern in ansteigender Ordnung.
Auf diese Weise wird die Reihenfolge für das Einlesen der Daten im allgemeinen durch die Reihung der Daten innerhalb der Datenliste und durch die Anordnung der DATA-Statements selbst bestimmt.

Die Daten in der Datenliste müssen mit dem Variablentyp übereinstimmen, dem sie zugeordnet werden. Wenn also im READ-Statement eine numerische Variable angegeben ist, muß die entsprechende Stelle im DATA-Statement mit einer numerischen Konstanten belegt sein. Ist eine String-Variable angegeben, muß die entsprechende Stelle im DATA-Statement eine String-Konstante aufweisen. Bedenken Sie, daß eine Zahl ein gültiger String ist: wenn also eine String-Konstante erforderlich ist, kann an der entsprechenden Stelle im DATA-Statement auch eine Zahl sein.

Bei Anwendung von String-Konstanten im Data-Statement kann man den String in Anführungszeichen setzen. Die Anführungszeichen *müssen* gesetzt werden, wenn der String, den Sie eingeben wollen, ein Komma, ein einleitendes Anführungszeichen, oder einleitende bzw. abschließende Leerstellen enthält.

Wenn die Liste der String-Konstanten im DATA-Statement nebeneinanderliegende Kommata enthält, nimmt der Computer an, daß Sie einen Null-String eingeben wollen (einen String ohne Zeichen). Im Beispiel rechts weist das DATA-Statement in Zeile 110 zwei Kommata nebeneinander auf. Somit wird für B$ ein Null-String zugeordnet, wie Sie beim Ablauf des Programms verfolgen können.

Wenn bei Durchführung eines READ-Statements die Variablenliste mehr Namen enthält als Werte in den DATA-Statements vorhanden sind, wird die Meldung "DATA ERROR" (Datenfehler) auf dem Bildschirm angezeigt und der Programmablauf wird unterbrochen. Wird eine numerische Konstante gelesen, die eine Bereichsunterschreitung verursacht, wird ihr Wert durch Null ersetzt, und das Programm setzt - ohne Warnung - den Ablauf normal fort.

Wenn eine numerische Konstante gelesen wird, die einen Überlauf bewirkt, wird die Nachricht „WARNING: NUMBER TOO BIG" (Warnung: zu große Zahl) angezeigt, der Wert der Konstanten durch die entsprechende Computergrenze ersetzt, und das Programm fortgesetzt. Informationen zu Kapazitätsüberlauf, -unterlauf und numerischen Grenzen finden Sie im Abschnitt „Numerische Konstante".

Beispiele:

```
>NEW

>100 FOR I=1 TO 5
>110 READ A,B
>120 PRINT A;B
>130 NEXT I
>140 DATA 2,4,6,7,8
>150 DATA 1,2,3,4,5
>160 END
>RUN
  2  4
  6  7
  8  1
  2  3
  4  5

 ** DONE **
```

```
>NEW

>100 READ A$,B$,C,D
>110 PRINT A$:B$:C:D
>120 DATA HELLO,"JONES, MARY"
 ,28,3.1416
>130 END
>RUN
 HELLO
 JONES, MARY
  28
  3.1416

 ** DONE **
```

```
>NEW

>100 READ A$,B$,C
>110 DATA HI,,2
>120 PRINT "A$ IS ";A$
>130 PRINT "B$ IS ";B$
>140 PRINT "C IS ";C
>150 END
 RUN
 A$ IS HI
 B$ IS
 C IS  2

 ** DONE **
```

```
>NEW
>100 READ A,B
>110 DATA 12E-135
>120 DATA 36E142
>130 PRINT :A:B
>140 READ C
>150 END
>RUN

 * WARNING:
   NUMBER TOO BIG IN 100

  0
  9.99999E+**

 * DATA ERROR IN 140

>□
```

RESTORE

RESTORE [*Zeilennummer*]

Weitere Informationen zur Anwendung von RESTORE finden Sie im Abschnitt "Dateiverwaltung".
Das RESTORE-Statement weist den Computer an, welches DATA-Statement zusammen mit dem nächsten READ-Statement zu verwenden ist. Wenn RESTORE ohne Zeilennummer angegeben und das nächste READ-Statement durchgeführt wird, werden die Werte, beginnend mit dem ersten DATA-Statement, im Programm zugeordnet.

Folgt auf RESTORE die Zeilennummer eines DATA-Statements, dann werden bei der Durchführung des nächsten READ-Statements die Werte, beginnend mit dem ersten Datenpunkt, in dem DATA-Statement zugeordnet, das durch die Zeilennummer festgelegt wurde.

Wenn Sie das Programm rechts in der Beispiel-Spalte verfolgen, können Sie die Wechselwirkung der Statements READ, DATA und RESTORE verfolgen. In Zeile 120 beginnt der Computer, den Variablen A und B aus dem DATA-Statement mit der niedrigsten Zeilennummer, 180, die Werte zuzuordnen. Nach der ersten READ-Anweisung erhält demnach A den Wert 2 und B den Wert 4.

Bei der nächsten Durchführung des READ-Statements werden immer noch die Daten aus Zeile 180 aufgenommen, und die Zuordnung ist A = 6 und B = 8. Die dritte READ-Anweisung ordnet das letzte Element in Zeile 180 der Variablen A zu, und das erste Element in Zeile 190 der Variablen B; demnach ist A = 10 und B = 12. Das vierte READ-Statement, das letzte in der J-Schleife, setzt die Übernahme der Daten aus Zeile 190 fort; A ist also 14 und B ist 16. Achten Sie jedoch darauf, daß vor dem erneuten Durchlauf der I-Schleife der Computer in Zeile 160 auf ein RESTORE-Statement trifft, das ihn anweist, die Daten für das nächste READ-Statement aus dem Anfang von Zeile 190 zu entnehmen.

Der Computer schließt das Programm mit dem Lesen der Daten aus Zeile 190 und dann aus Zeile 200 ab.

Wenn die im RESTORE-Statement angegebene Zeilennummer kein DATA-Statement oder keine Programm-Zeilennummer ist, beginnt der Computer bei der Durchführung des nächsten READ-Statements mit dem ersten Data-Statement, dessen Zeilennummer höher ist als die angegebene.

Existiert kein DATA-Statement mit einer Zeilennummer größer oder gleich der angegebenen, entsteht bei der Durchführung des nächsten READ-Statements ein Datenmangel und die Nachricht "DATA ERROR" (Datenfehler) wird angezeigt.

Ist die angegebene Zeilennummer größer als die höchste Zeilennummer im Programm, unterbricht das Programm den Ablauf und die Nachricht "DATA ERROR IN xx" (Datenfehler in xx) wird angezeigt.

Beispiele:

```
>NEW

>100 FOR I=1 TO 5
>110 READ X
>120 RESTORE
>130 PRINT X;
>140 NEXT I
>150 DATA 10,20,30
>160 END
>RUN
  10  10  10  10  10
 ** DONE **
```

```
>NEW

>100 FOR I=1 TO 2
>110 FOR J=1 TO 4
>120 READ A
>130 PRINT A;
>140 NEXT J
>150 RESTORE 180
>160 NEXT I
>170 DATA 12,33,41,26,42,50
>180 DATA 10,20,30,40,50
>190 END
>RUN
  12  33  41  26  10  20  30
  40
 ** DONE **
```

```
>NEW

>100 FOR I=1 TO 2
>110 FOR J=1 TO 4
>120 READ A,B
>130 PRINT A;B;
>140 NEXT J
>150 PRINT
>160 RESTORE 190
>170 NEXT I
>180 DATA 2,4,6,8,10
>190 DATA 12,14,16,18
>200 DATA 20,22,24,26
>210 END
>RUN
  2  4  6  8  10  12  14  16
  12  14  16  18  20  22  24
  26

 ** DONE **
```

PRINT

PRINT [*Druckliste*]

(Informationen über das PRINT-Statement in Verbindung mit Dateien finden Sie im Abschnitt "Dateiverarbeitung").
Mit dem PRINT-Statement können Sie Zahlen und Strings auf dem Bildschirm ausdrucken lassen.
Die Druckliste besteht aus:

- **Elementen** – numerischen Ausdrücken und Stringausdrücken, die auf dem Bildschirm angezeigt werden, und Tab-Funktionen, die die Schreibstellen steuern (ähnlich wie die TAB-Taste auf der Schreibmaschine);
- **Separatoren (Trennzeichen)** – die Interpunktion zwischen den Druckelementen (Kommata, Doppelpunkte, Semikolon), die als Indikatoren für die Position der Daten auf der Druckzeile dienen. Wenn der Computer ein PRINT-Statement ausführt, werden die Werte der Ausdrücke in der Druckliste in der Folge von links nach rechts auf dem Bildschirm angezeigt, entsprechend den Angaben durch die Druckseparatoren und die Tab-Funktionen.

Ausdrucken von Strings

String-Ausdrücke in der Druckliste werden als solche ausgegeben. Vor oder nach dem String werden keine Leerstellen eingefügt. Wenn Sie den Ausdruck einer Leerstelle vor oder nach dem String wünschen, müssen Sie diese in den String selbst einbeziehen oder gesondert mit Anführungszeichen einfügen.

Ausdrucken von Zahlen

Numerische Ausdrücke in der Druckliste werden aufgezeigt, um den Ausdruck eines numerischen Resultats zu bewirken. Positive Zahlen werden mit einer vorangehenden Leerstelle (anstelle des positiven Vorzeichens) gedruckt. Beim Ausdruck von negativen Zahlen wird das Minuszeichen vorangestellt. Alle Zahlen werden mit anschließendem Leerzeichen gedruckt.

In Extended Basic wird das PRINT-Statement noch durch PRINT ...USING erweitert. Man kann dann auch das Format kontrollieren.*

* separat als Zubehör

Beispiele:

```
>NEW

>100 READ A,B
>110 RESTORE 130
>120 PRINT A;B
>130 READ C,D
>140 PRINT C;D
>150 DATA 26.9,34.67
>160 END
>RUN
  26.9   34.67
  26.9   34.67

 ** DONE **
```

```
>NEW

>100 N$="JOAN"
>110 M$="HI"
>120 PRINT M$;N$
>130 PRINT M$&" "&N$
>140 PRINT "HELLO ";N$
>150 END
>RUN
 HIJOAN
 HI JOAN
 HELLO JOAN

 ** DONE **
```

```
>NEW

>100 LET A=10.2
>110 B=-30.5
>120 C=16.7
>130 PRINT A;B;C
>140 PRINT A+B
>150 END
>RUN
 10.2 -30.5  16.7
-20.3

 ** DONE **
```

PRINT

Druckseparatoren (Trennzeichen)

Jede mit dem PRINT-Statement dargestellte Bildschirmzeile hat 28 Schreibstellen. Jede Zeile wird in zwei Zonen zu je 14 Zeichen eingeteilt. Durch Druckseparatoren (Trennzeichen) und durch die Tab-Funktion können Sie die Position der auf dem Bildschirm angezeigten Zeichen steuern.

Es gibt drei Arten von Druckseparatoren: das Semikolon, den Doppelpunkt und das Komma. Mindestens ein Druckseparator muß zwischen zwei aneinanderliegenden Elementen in der Druckliste angegeben werden. Es ist möglich, mehrere Trennzeichen nebeneinander zu verwenden; sie werden dann von links nach rechts ausgewertet.

Der Druckseparator Semikolon bewirkt, daß Zeichen ohne Leerstellen ausgedruckt werden.

Alle Zahlen werden allerdings generell mit einer nachfolgenden Leerstelle ausgedruckt, unabhängig vom Typus des verwendeten Trennzeichens.

Das Trennzeichen Doppelpunkt bewirkt, daß das nächste Element am Anfang der nächsten Zeile ausgedruckt wird.

Die Druckzeilen sind in zwei Zonen aufgeteilt. Die erste Zone beginnt bei Zeichen 1 (Spalte 1), die zweite bei Zeichen 15 (Spalte 15). Wenn der Computer den Druckseparator Komma auswertet, wird das nächste Element zu Beginn der nächsten Zone ausgedruckt. Wenn bei der Auswertung des Komma-Trennzeichens bereits die zweite Zone erreicht ist, wird das nächste Druckelement in die nächste Zeile gesetzt.

Beispiele.

```
>NEW

>100 A=-26
>110 B=-33
>120 C$="HELLO"
>130 D$="HOW ARE YOU?"
>140 PRINT A;B;C$;D$
>150 END
>RUN
 -26 -33 HELLOHOW ARE YOU?

 ** DONE **
```

```
>NEW

>100 A=-26
>110 B$="HELLO"
>120 C$="HOW ARE YOU?"
>130 PRINT A:B$:C$
>140 END
>RUN
 -26
 HELLO
 HOW ARE YOU?

 ** DONE **
```

```
>NEW

>100 A$="ZONE 1"
>110 B$="ZONE 2"
>120 PRINT A$,B$
>130 PRINT A$:,B$,A$
>140 END
>RUN
 ZONE 1          ZONE 2
 ZONE 1
                 ZONE 2
 ZONE 1

 ** DONE **
```

PRINT

Tab-Funktionen

Mit der Tab-Funktion können Sie ähnlich wie bei der Schreibmaschine Ihren Ausdruck tabellieren.

TAB (numerischer Ausdruck)

Mit der Angabe der Spaltenzahl definieren Sie die Stelle der Bildschirmzeile, an der die Druckzeile stehen soll. Der Computer wertet die Spaltenzahl aus und rundet auf die nächste ganze Zahl n. Ist n kleiner 1, wird der Wert durch 1 ersetzt. Ist n größer als 28, dann wird n wiederholt um 28 reduziert, bis $1 \leq n \leq 28$.
Ist die Zahl der Zeichen, die bereits in der momentanen Zeile gedruckt sind, kleiner oder gleich n, dann wird das nächste Element, beginnend in Schreibstelle n, ausgedruckt. Wenn die Anzahl der bereits gedruckten Zeichen auf der momentanen Zeile größer als n ist, wird das nächste Element in der nächsten Zeile, beginnend mit Schreibstelle n, ausgedruckt.

Beachten Sie, daß die Tab-Funktion ein Druckelement ist, und daß daher ein Druckseparator vorangestellt werden muß, es sei denn, sie ist das erste Element in der Druckliste. Der Tab-Funktion muß ebenfalls ein Druckseparator folgen, ausgenommen, sie ist das letzte Element in der Druckliste. Das Trennzeichen vor der Tab-Funktion wird vorher, das Trennzeichen nach der Tab-Funktion im Anschluß an die Tab-Funktion ausgewertet. Wir empfehlen daher als Trennzeichen vor und nach der Tab-Funktion ein Semikolon, um optimale Ergebisse zu erzielen.

Beispiele:

Im Programm rechts führt der Computer folgende Maßnahmen durch:

- Zeile 120 - druckt A, rückt zur Schreibstelle 15, druckt B;
- Zeile 130 - druckt A, rückt zur nächsten Zone (in diesem Fall Schreibstelle 15 der momentanen Bildschirmzeile), druckt B;
- Zeile 140 - druckt A, rückt zur Schreibstelle 15 nach der Angabe in der Tab-Funktion, rückt dann wegen des Kommas zur nächsten Druckzone (in diesem Fall Schreibstelle 1 der nächsten Bildschirmzeile), druckt B;
- Zeile 150 - rückt zur Schreibstelle 5, druckt A, rückt zur Schreibstelle 6 der nächsten Zeile (da Schreibstelle 6 der momentanen Zeile bereits passiert war, als A gedruckt wurde), druckt B;
- Zeile 160 - druckt A, subtrahiert 28 von 43, um die Tab-Funktion innerhalb der zulässigen Schreibstellen zu beginnen, rückt zur Schreibstelle 15 (43 — 28 = 15), druckt B.

```
>NEW

>100 A=326
>110 B=79
>120 PRINT A;TAB(15);B
>130 PRINT A,B
>140 PRINT A;TAB(15),B
>150 PRINT TAB(5);A;TAB(6);B
>160 PRINT A;TAB(43);B
>170 END
 RUN
  326             79
  326             79
  326
  79
      326
       79
  326             79

 ** DONE **
```

DISPLAY

DISPLAY [*Druckliste*]

Das DISPLAY-Statement (Anzeige-Statement) ist mit dem PRINT-Statement identisch, wenn es zum Ausdrucken von Elementen auf dem Bildschirm verwendet wird. Das DISPLAY-Statement eignet sich nicht zum Schreiben auf einem Peripherie-Gerät.

Für das DISPLAY-Statement gelten dieselben Regeln wie für das PRINT-Statement.

In Extended Basic wird auch das DISPLAY-Statement durch DISPLAY ... USING erweitert.*

* separat als Zubehör

Beispiele:

```
>NEW

>100 A=35.6
>110 B$="HI!!"
>120 C=49.7
>130 PRINT B$:A;C
>140 DISPLAY B$:A;C
>150 END
>RUN
 HI!!
  35.6  49.7
 HI!!
  35.6  49.7

 ** DONE **
```

Farbgraphiken und akustische Signale

Einführung

Ihr Home Computer-System enthält spezielle Unterprogramme für Farbgraphiken, akustische Signale usw., die es im allgemeinen bei BASIC nicht gibt.

Wenn Sie eines dieser Unterprogramme benutzen wollen, rufen Sie seinen Titel auf und fügen noch einige wenige Spezifikationen hinzu. Das Unterprogramm wird dann übernommen und ermöglicht Ihnen die Erzeugung von Tönen, zaubert Farben auf den Bildschirm sowie besondere graphische Zeichen. Alle Unterprogramme können im Befehlsmodus und in Programmen verwendet werden.

Die eingebauten Unterprogramme lassen sich entsprechend ihrer Funktion gruppieren:

- INPUT-Unterprogramme (Eingabe-Unterprogramme)
  - GCHAR, JOYST, KEY
- OUTPUT-Unterprogramme (Ausgabe-Unterprogramme)
- CLEAR, HCHAR, VCHAR, SOUND, SCREEN
- INTERNAL-Unterprogramme (Interne Unterprogramme)
  - CHAR, COLOR (deren Ergebnisse werden erst bei Anwendung einer OUTPUT-Operation auf dem Schirm sichtbar).

Extended Basic erweitert diese Unterprogramme noch durch weitere Programme, die den Programmierkomfort und die Nutzungsmöglichkeiten des TI 99/4A enorm vergrößern (z. B. CHARPAT; CHARSET; COINC; LOCATE; MAGNIFY; MOTION; PATTERN; SAY**; SPRITE u. a.).*

* separat als Zubehör ** in Verbindung mit dem TI-Sprachsynthesizer

CLEAR-Unterprogramm

CALL CLEAR
(Aufruf CLEAR)

Das CLEAR-Unterprogramm wird zum Löschen des gesamten Bildschirms verwendet. Wenn das CLEAR-Unterprogramm aufgerufen wird, belegt das Leerzeichen (ASCII-Kode 32) alle Schreibstellen auf dem Bildschirm.

Bei Ablauf des Programms auf der rechten Seite wird der Bildschirm vor der Durchführung der PRINT-Statements gelöscht.

Beispiele:

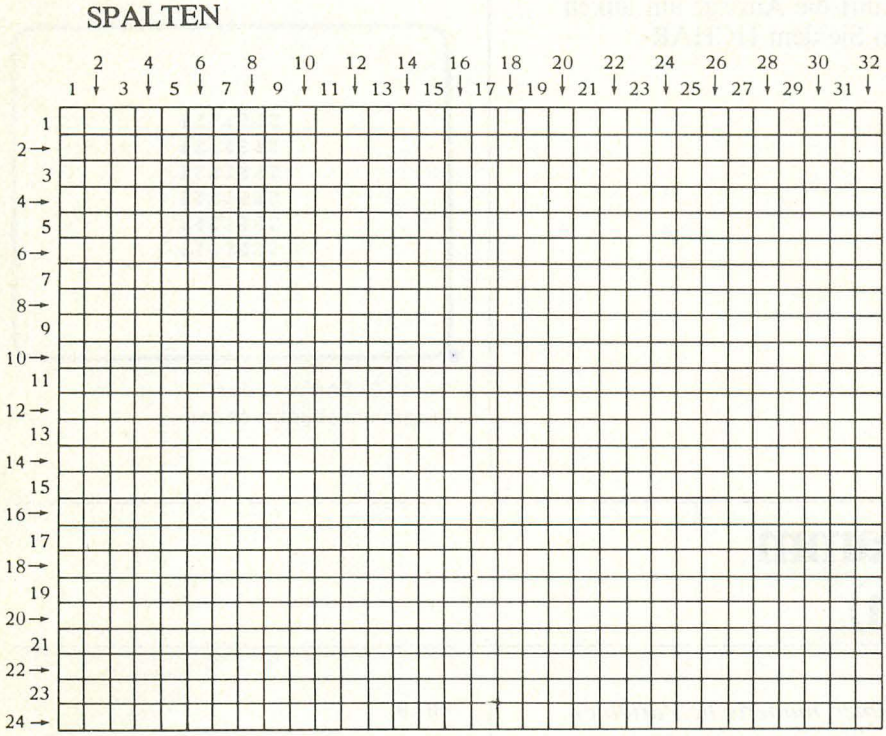
```
>PRINT "HELLO THERE!"
 HELLO THERE!
>CALL CLEAR

 --Bildschirm  wird  gelöscht
```

```
>NEW

>100 CALL CLEAR
>110 PRINT "HELLO THERE!"
>120 PRINT "HOW ARE YOU?"
>130 END
>RUN

 --Bildschirm  wird  gelöscht

 HELLO THERE!
 HOW ARE YOU?

 ** DONE **
```

HCHAR-Unterprogramm (Zeichenwiederholung horizontal)

CALL HCHAR *(Zeilennummer, Spaltennummer, Zeichenkode, Anzahl der Wiederholungen)*

Das HCHAR-Unterprogramm setzt an beliebiger Stelle auf dem Bildschirm ein Zeichen und wiederholt es wahlweise in horizontaler Richtung. Die Zeilennummer und die Spaltennummer lokalisieren den Startpunkt auf dem Bildschirm. Zeilennummer, Spaltennummer, Zeichenkode und Anzahl der Wiederholungen sind numerische Ausdrücke.



Wenn die Auswertung einer dieser numerischen Ausdrücke zu einem nicht-ganzzahligen Wert führt, wird das Resultat auf eine ganze Zahl gerundet.

Die gültigen Bereiche sind nachstehend angegeben:

Wert	Bereich
Zeilennummer	1 bis 24 inklusive
Spaltennummer	1 bis 32 inklusive
Zeichenkode	0 bis 32767 inklusive
Anzahl der Wiederholungen	0 bis 32767 inklusive

Da die Spalten 1, 2, 31 und 32 vielleicht nicht mehr im Anzeigebereich Ihres Bildschirms liegen, verwenden Sie in diesem Fall nur die Spaltennummern 3 bis 30.

Obwohl Sie einen Wert von bis zu 32767 für den Zeichenkode angeben können, wandelt der Computer diesen Wert in einen zwischen 0 und 255 um. Die Zeichenkodes zwischen 32 und 127 sind als die Standard-ASCII-Zeichenkodes definiert. Die Zeichenkodes von 128 bis 159 können bei Anwendung des CHAR-Unterprogramms definiert werden. Wenn Sie ein undefiniertes Zeichen für den Zeichenkode spezifizieren, erhalten Sie als Resultat immer das, was zur Zeit des Aufrufs von Unterprogramm HCHAR im Speicher war.

Beispiele:

```
>CALL CLEAR

 --Bildschirm wird gelöscht

>CALL HCHAR(10,1,72,50)
```

```
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
HHHHHHHHHHHHHHHHH

>CALL HCHAR(10,1,72,50)
```

```
>NEW

>100 CALL CLEAR
>110 FOR I=9 TO 15
>120 CALL HCHAR(I,13,36,6)
>130 NEXT I
>140 GOTO 140
>RUN

 --Bildschirm wird gelöscht
```

```
$$$$$$
$$$$$$
$$$$$$
$$$$$$
$$$$$$
$$$$$$
$$$$$$
```

(Durch CLEAR wird das Programm abgebrochen)

VCHAR-Unterprogramm (Zeichenwiederholung vertikal)

CALL VCHAR *(Zeilennummer, Spaltennummer, Zeichenkode, Anzeige der Wiederholungen)*

Die Durchführung des VCHAR-Unterprogramms entspricht weitgehend der HCHAR-Routine, nur werden die Zeichen vertikal und nicht horizontal wiederholt. Der Computer beginnt die Anzeige des Zeichens mit der angegebenen Startposition und setzt die Anzeige bis zum unteren Ende des Bildschirms fort.

Wenn die untere Zeile erreicht ist, wird die Wiederholung des Zeichens in der nächsten rechten Spalte oben weitergeführt. Ist dann der rechte Rand des Bildschirms erreicht, fährt die Anzeige am linken Rand fort. (Weitere Einzelheiten entnehmen Sie dem HCHAR-Unterprogramm).

Beispiele:

```
>NEW

>100 CALL CLEAR
>110 FOR I=13 TO 18
>120 CALL VCHAR(9,I,36,6)
>130 NEXT I
>140 GOTO 140
>RUN

 -- Bildschirm wird gelöscht
```

```
$$$$$$$
$$$$$$$
$$$$$$$
$$$$$$$
$$$$$$$
$$$$$$$
```

Durch CLEAR wird das Programm abgebrochen)

GCHAR-Unterprogramm (Ablesen von Zeichen)

CALL GCHAR *(Zeilennummer, Spaltennummer, numerische Variable)*

Das GCHAR-Unterprogramm erlaubt das Ablesen eines Zeichens von beliebiger Stelle auf dem Bildschirm. Die Position des gewünschten Zeichens wird durch die Zeilennummer und durch die Spaltennummer beschrieben. Der Computer setzt den numerischen ASCII-Kode des geforderten Zeichens in die numerische Variable ein, die Sie im CALL GCHAR Statement spezifizieren.

Zeilennummer und Spaltennummer sind numerische Ausdrücke. Wenn ihre Auswertung zu einem nicht-ganzzahligen Wert führt, wird das Resultat auf eine ganze Zahl gerundet. Mit einem Wert von 1 für die Zeilennummer wird die obere Zeile des Bildschirms angegeben. Mit einem Wert von 1 für die Spaltennummer wählt man den linken Bildschirmrand. Siehe Gitter unter HCHAR.

```
>NEW

>100 CALL CLEAR
>110 CALL HCHAR(1,1,36,768)
>120 CALL GCHAR(5,10,X)
>130 CALL CLEAR
>140 PRINT X
>150 END
>RUN

 --Bildschrim wird gelöscht

 --Bildschirm wird mit $$$
   aufgefüllt (Kode 36)

 --Bildschirm wird gelöscht

  36

 ** DONE **
```

COLOR-Unterprogramm

CALL COLOR *(Zeichensatz-Nummer, Vordergrund-Farbkode, Hintergrund-Farbkode)*

Das COLOR-Unterprogramm ermöglicht auf dem Bildschirm farbige Zeichen. (Zur Veränderung der Bildschirmfarbe selbst siehe das SCREEN-Unterprogramm. Die Zeichensatznummer (character set number), der Vordergrund-Farbkode (foreground color code) und der Hintergrund-Farbkode (background color code) sind numerische Ausdrücke.

Jedes auf dem Bildschirm angezeigte Zeichen hat zwei Farben. Die Farbe der Punkte, die das Zeichen bilden, ist die Vordergrundfarbe. Die Farbe auf dem Rest der Schreibstelle ist die Hintergrundfarbe. Beim Video-Computer stehen Ihnen 16 Farben zur Verfügung. Ihre Eingaben für Vorder- und Hintergrundfarbe müssen also einen Wert von 1 bis 16 haben.

Hier die Farbkodes:

Farbkode	Farbe
1	transparent
2	schwarz
3	mittelgrün
4	hellgrün
5	dunkelblau
6	hellblau
7	dunkelrot
8	kornblumenblau
9	mittelrot
10	hellrot
11	dunkelgelb
12	hellgelb
13	dunkelgrün
14	magentarot
15	grau
16	weiß

Gibt man als Farbe transparent (Kode 1) an, scheint bei der Anzeige eines Zeichens die augenblickliche Bildschirmfarbe durch. Bis zur Durchführung von CALL COLOR ist die Standard-Vordergrundfarbe für alle Zeichen schwarz (Kode 2) und die Standard-Hintergrundfarbe transparent (Kode 1). Bei einer Stoppstelle (siehe BREAK) werden alle Zeichen auf die Standardfarben zurückgesetzt.

Beispiele:

```
>NEW

>100 CALL CLEAR
>110 INPUT "FOREGROUND?":F
>120 INPUT "BACKGROUND?":B
>130 CALL CLEAR
>140 CALL COLOR(2,F,B)
>150 CALL HCHAR(12,3,42,28)
>160 GO TO 110
>RUN

 --Bildschirm wird gelöscht

 FOREGROUND?2
 BACKGROUND?14

 --Bildschirm wird gelöscht
```

(28 schwarze Sternchen auf einem magenta Hintergrund)

```
****************************

 FOREGROUND?
```

(Durch CLEAR wird das Programm abgebrochen)

```
>NEW

100 CALL CLEAR
110 CALL COLOR(2,1,7)
120 CALL HCHAR(12,3,42,28)
130 GOTO 130
>RUN

 --Bildschirm wird gelöscht
```

(schwarze Sternchen mit einem dunkelroten Hintergrund auf grünen Bildschirm)

```
****************************
```

(Durch CLEAR wird das Programm abgebrochen)

COLOR-Unterprogramm

Um CALL COLOR anzuwenden, müssen Sie auch angeben (spezifizieren), zu welchem der 16 Zeichensätze das Zeichen gehört, das Sie gerade drucken. Die Liste der ASCII-Zeichenkodes für die Standardzeichen finden Sie auf Seite 99. Das Zeichen wird in der angegebenen Farbe ausgewiesen, wenn Sie CALL HCHAR oder CALL VCHAR anwenden. Nachstehend sehen Sie eine Liste der Zeichensatz-Nummer:

Satznummer	Zeichenkodes
1	32-39
2	40-47
3	48-55
4	56-63
5	64-71
6	72-79
7	80-87
8	88-95
9	96-103
10	104-111
11	112-119
12	120-127
13	128-135
14	136-143
15	144-151
16	152-159

Beachten Sie, daß alle 24 Reihen und 32 Spalten mit dem Leerzeichen belegt werden, bis Sie in einige dieser Schreibstellen andere Zeichen plazieren. Wenn Sie Zeichensatz 1 im CALL COLOR Statement anwenden, werden alle Leerzeichen auf dem Bildschirm auf die angegebene Hintergrundfarbe abgeändert, da das Leerzeichen im Satz 1 enthalten ist. Diese Veränderung wird mit dem Programm rechts demonstriert.

Beispiele:

```
>NEW

100 CALL CLEAR
110 CALL COLOR(1,16,14)
120 CALL VCHAR(1,15,35,24)
130 GOTO 130
>RUN
```

--Bildschirm wird gelöscht

--24 weiße # mit magenta Hintergrund auf einem grünen Bildschirm

```
#
#
#
#
#
#
#
#
#
#
#
#
```

--Beachten Sie, daß die Bildschirmfarbe nur am oberen und unteren Rand des Bildschirms erscheint

(Durch CLEAR wird das Programm abgebrochen)

SCREEN-Unterprogramm

CALL SCREEN *(Farbkode)*

Das SCREEN-Unterprogramm erweitert die Graphikmöglichkeiten des Home-Computers durch die Änderung der Bildschirmfarbe.
Die Standard-Bildschirmfarbe während eines Programmablaufs ist hellgrün (Farbkode 4).

Der Farbkode ist ein numerischer Ausdruck mit einem Wert zwischen 1 und 16. Siehe Tabelle CALL COLOR.

Bei Durchführung von CALL SCREEN ändert sich der gesamte Bildschirm-Hintergrund auf die im Farbkode angegebene Farbe. Alle Zeichen auf dem Bildschirm bleiben gleich, es sei denn, Sie haben für sie einen transparenten Vorder- oder Hintergrund angegeben. In diesem Fall scheint die Bildschirmfarbe durch den transparenten Vorder -oder Hintergrund.

Der Bildschirm wird auf kornblumenblau (Kode 8) eingestellt, wenn das Programm wegen einer Stoppstelle unterbricht oder beendet wird.

Wenn Sie das Programm nach einer Stoppstelle wieder mit CONTINUE fortsetzen, stellt sich der Bildschirm wieder auf die Standardfarbe hellgrün um.

Beispiele:

```
>NEW

>100 CALL CLEAR
>110 INPUT "SCREEN COLOR?":S
>120 INPUT "FOREGROUND?":F
>130 INPUT "BACKGROUND?":B
>140 CALL CLEAR
>150 CALL SCREEN(S)
>160 CALL COLOR(2,F,B)
>170 CALL HCHAR(12,3,42,28)
>180 GOTO 110
>RUN

 --Bildschirm wird gelöscht

 SCREEN COLOR?7
 FOREGROUND?13
 BACKGROUND?16

 --Bildschirm wird gelöscht

 --28 dunkelgrüne Sternchen
    mit weißem Hintergrund auf
    dunkelrotem Bildschirm
```

```
****************************

  SCREEN COLOR?
```

(Durch CLEAR wird das Programm abgebrochen)

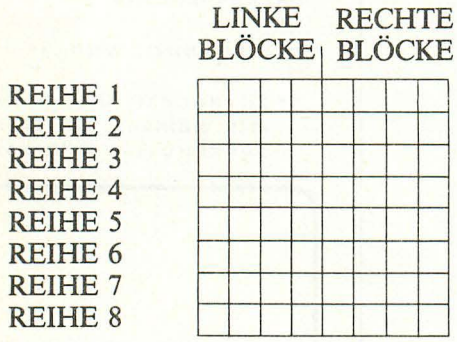
CHAR-Unterprogramm (Zeichendefinition)

CALL CHAR *(Zeichenkode, "Muster-Kodierer")*

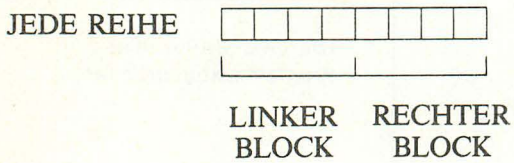
Mit dem CHAR-Unterprogramm (Zeichen-Unterprogramm) können Sie Ihre eigenen speziellen graphischen Zeichen definieren. Sie haben die Möglichkeit, den Standard-Zeichensatz (ASCII-Kodes 32 bis 127) neu zu definieren, und zusätzliche Zeichen mit Kodes von 128 bis 159 festzulegen.

Der Zeichenkode gibt den Kode des Zeichens an, das Sie definieren wollen. Er muß ein numerischer Ausdruck mit einem Wert von 32 bis 159 inklusive sein. Liegt das Zeichen, das Sie im Bereich 128 bis 159 definieren und reicht der freie Speicherplatz für die Definition nicht aus, endet das Programm mit der Fehlermeldung "MEMORY FULL" (Speicher belegt).

Die Muster-Kodierung ist ein 16 Zeichen umfassender String-Ausdruck, der die Form des Zeichens angibt, das Sie in Ihrem Programm verwenden wollen. Der String-Ausdruck ist eine kodierte Darstellung der 64 Punkte, die eine Schreibstelle auf dem Bildschirm ausmachen. Diese 64 Punkte bilden ein Rasterfeld von 8 mal 8 Positionen, wie unten in starker Vergrößerung gezeigt wird.



Jede Reihe ist in zwei Blöcke zu jede 4 Punkten aufgeteilt:



Beispiele:

```
>NEW

>100 CALL CLEAR
>110 CALL CHAR(33,"FFFFFFFFFF
 FFFFFF")
>120 CALL COLOR(1,9,6)
>130 CALL VCHAR(12,16,33)
>140 GOTO 140
>RUN
```

-- Bildschirm wird gelöscht

-- mittelroter Block mit hellblauem Hintergrund auf hellgrünem Bildschirm

(Durch CLEAR wird das Programm abgebrochen)

CHAR-Unterprogramm

Jedes Zeichen im String-Ausdruck beschreibt das Punktemuster in einem Block einer Reihe. Die Reihen werden von links nach rechts und von oben nach unten beschrieben. Das heißt, die ersten beiden Zeichen im String geben das Muster für Reihe 1 in dem Punkte-Rasterfeld an, die nächsten beiden beschreiben Reihe 2 usw.

Die Zeichen werden dadurch erzeugt, daß man einige Punkte "eingeschaltet" und andere "ausgeschaltet" läßt. Beim Leerzeichen (Kode 32) sind alle Punkte "ausgeschaltet". Wenn man alle Punkte "einschaltet", entsteht ein fester Block (■).

Alle Standardzeichen sind automatisch so eingestellt, daß sie die entsprechenden Punkte "einschalten". Um ein neues Zeichen zu schaffen, müssen Sie den Computer anweisen, welche Punkte in jedem der 16 Blöcke, die das Zeichen enthalten, einzuschalten sind, und welche ausgeschaltet bleiben müssen. Im Computer wird ein Binärkode verwendet, um anzugeben, welche Punkte innerhalb eines bestimmten Blocks ein- bzw. ausgeschaltet sind. Zur Kontrolle der Ein-/Aus-Bedingung setzt man jedoch ein "Schnellverfahren", die Hexadezimalmethode, ein, die sich aus Zahlen und Buchstaben zusammensetzt. Die nachstehende Tabelle enthält alle möglichen Ein-/Aus-Bedingungen für die Punkte innerhalb eines gegebenen Blocks und die Hexadezimaldarstellung für jede Möglichkeit.

Blöcke	Binärkode 0 = OFF (aus) 1 = ON (ein)	Hexadezimalkode
	0000	0
	0001	1
	0010	2
	0011	3
	0100	4
	0101	5
	0110	6
	0111	7
	1000	8
	1001	9
	1010	A
	1011	B
	1100	C
	1101	D
	1110	E
	1111	F

Wichtig!
Die Hexadezimalkodes A, B, C, D, E und F müssen als Großbuchstaben eingegeben werden.

CHAR-Unterprogramm

Um das unten dargestellte Punktemuster zu beschreiben, müssen Sie für CALL CHAR den folgenden String kodieren: "1898FF3D3C3CE404"

	LINKE BLOCKS	RECHTE BLOCKS	BLOCK KODES
REIHE 1			18
REIHE 2			98
REIHE 3			FF
REIHE 4			3D
REIHE 5			3C
REIHE 6			3C
kEIHE 7			E4
REIHE 8			04

Wenn der String-Ausdruck weniger als 16 Zeichen umfaßt, geht der Computer davon aus, daß die restlichen Zeichen null sind. Ist der String länger als 16 Zeichen, ignoriert der Computer die Überschreitung.

Bedenken Sie, daß CALL CHAR das Zeichen lediglich definiert. Zur Anzeige des Zeichens auf dem Bildschirm müssen Sie CALL HCHAR, CALL VCHAR, PRINT oder DISPLAY anwenden. Wird CALL CHAR durchgeführt, wird jedes bereits angezeigte Zeichen mit dem gleichen Kode in das neue Zeichen umgeformt.

Beispiele:

```
>NEW

>100 CALL CLEAR
>110 A$="1898FF3D3C3CE404"
>120 B$="1819FFBC3C3C2720"
>130 CALL CHAR (128, A$)
>140 CALL CHAR (129, B$)
>150 CALL COLOR(9,7,12)
>160 CALL VCHAR (12, 16, 128)
>170 FOR DELAY=1 TO 500
>180 NEXT DELAY
>190 CALL VCHAR (12, 16, 129)
>200 FOR DELAY=1 TO 500
>210 NEXT DELAY
>220 GOTO 140
>230 END
>RUN

 --Bildschirm wird gelöscht

 --Das Zeichen bewegt sich vor-
   und rückwärls
```

(Durch CLEAR wird das Programm abgebrochen)

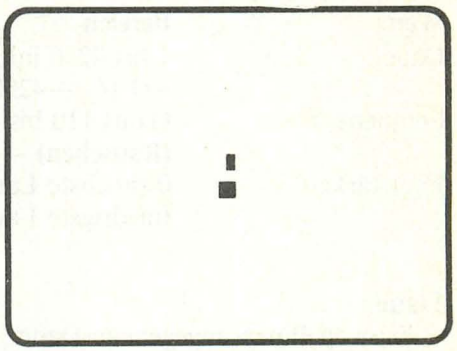
```
>NEW

>100 CALL CLEAR
>110 CALL CHAR (128, „0103070F1F
 3F7FFF")
>120 PRINT CHR $ (128)
>130 END
>RUN

 --Bildschirm wird gelöscht

 --◢

 ** DONE **
```

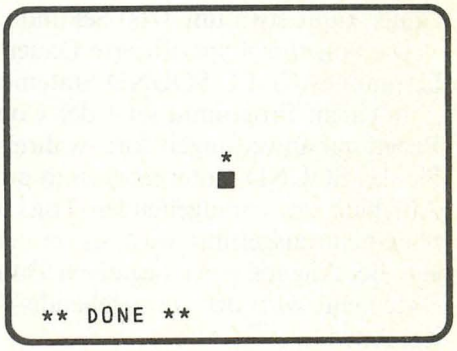
CHAR-Unterprogramm

Bei einer Programmunterbrechung wegen einer Stoppstelle, werden die Zeichen, die die Kodes 32 bis 127 neu definieren, in ihre normale Darstellung zurückgeführt. Die Zeichen mit den Kodes 128 bis 159 bleiben unverändert. Ist das Programm abgeschlossen, oder wird es wegen eines Fehlers abgebrochen, werden alle neu definierten Zeichen in ihre Normaldarstellung zurückgeführt, und alle den Kodes 128 bis 159 zugeordneten Zeichen werden in die undefinierte Form zurückgesetzt.

Beispiele:

```
>NEW

>100 CALL CLEAR
>110 CALL CHAR(128,„FFFFFFFFFF
 FFFFFF")
>120 CALL CHAR(42,"0F0F0F0F0F
 0F0F0F")
>130 CALL HCHAR(12,17,42)
>140 CALL VCHAR(14,17,128)
>150 FOR DELAY=1 TO 350
>160 NEXT DELAY
>170 END
>RUN
 --Bildschirm wird gelöscht
```






```
>CALL HCHAR(24,5,42)
   *
```

SOUND-Unterprogramm

CALL SOUND *(Dauer, Frequenz 1, Lautstärke 1 [,Frequenz 2, Lautstärke 2] [,Frequenz 3, Lautstärke3] [,Frequenz 4, Lautstärke 4])*

Das SOUND-Unterprogramm weist den Computer an, Töne verschiedener Frequenzen zu erzeugen. Die von Ihnen beigegebenen Werte steuern 3 Merkmale des Tons:

- Dauer - die Länge des Tons
- Frequenz - Tonhöhe
- Lautstärke

Dauer, Frequenz und Lautstärke sind numerische Ausdrücke. Wenn die Auswertung eines dieser numerischen Ausdrücke einen nicht-ganzzahligen Wert ergibt, wird das Ergebnis auf eine ganze Zahl gerundet. Die gültigen Bereiche für jeden Wert sind in der folgenden Tabelle aufgelistet, und anschließend näher erläutert.

Wert	Bereich
Dauer	1 bis 4250 inklusive
	—1 bis —4250 inklusive
Frequenz	(Ton) 110 bis 44733 inklusive
	(Rauschen) —1 bis —8 inklusive
Lautstärke	0 (höchste Lautstärke) bis 30
	(niedrigste Lautstärke) inklusive

Dauer

Die von Ihnen angegebene Dauer wird in Millisekunden gemessen. Die Dauer reicht also von .001 bis 4.25 Sekunden. (Die tatsächliche Dauer kann etwa um 1/60 Sekunde variieren).

Die von Ihnen spezifizierte Dauer gilt für jeden Ton, der durch ein bestimmtes CALL SOUND Statement erzeugt wird.

In einem Programm setzt der Computer die Ausführung von Programmanweisungen fort, während ein Ton abgespielt wird. Wenn Sie das SOUND-Unterprogramm aufrufen, wartet der Computer den Abschluß des vorangehenden Tons ab, ehe das neue CALL SOUND Statement ausgeführt wird, es sei denn, Sie geben eine negative Dauer ein. Bei Angabe einer negativen Dauer im neuen CALL SOUND Statement, wird der vorangehende Ton gestoppt und der neue sofort begonnen.

Beispiele:

```
>CALL SOUND(100,294,2)

 --ein einzelner Ton erklingt
```

```
>NEW

>100 TONE=110
>110 FOR COUNT=1 TO 10
>120 CALL SOUND(-500,TONE,1)
>130 TONE=TONE+110
>140 NEXT COUNT
>150 END
>RUN

 -- Zehn Töne erklingen schnell
    hintereinander
 ** DONE **

>120 CALL SOUND(+500,TONE,1)
>RUN

 -- Zehn Töne erklingen langsam
    hintereinander
 ** DONE **
```

SOUND-Unterprogramm

Frequenz

Die von Ihnen spezifizierte Frequenz kann entweder ein Ton oder ein Rauschen sein. Die Töne, gemessen in Hertz (eine Schwingung pro Sekunde, Hz), können von einer niedrigsten Tonhöhe von 110 Hz bis zu einer maximalen Tonhöhe von 44733 Hz (weit über dem menschlichen Hörbereich) angegeben werden. (Die tatsächlich erzeugte Frequenz kann je nach ihrem Wert zwischen 0 und 10% schwanken). Die Frequenzen für einige übliche Musiknoten entnehmen Sie bitte der unten angegebenen Tabelle Musik-Tonfrequenzen.

Beispiele:

```
>CALL SOUND(1000,440,2)
 --ein einzelner Ton erklingt
>CALL SOUND(500,-1,2)
 --ein einzelnes Geräusch erklingt
```

Tabelle 7: Musik-Tonfrequenzen

Die folgende Tabelle gibt die Frequenzen (auf ganze Zahlen gerundet) für vier Oktaven der temperierten Tonskala (ein halber Schritt zwischen den Noten) an. Die Liste stellt zwar nicht den gesamten Tonbereich dar - auch nicht den musikalischen - kann aber für die Musikprogrammierung eine wertvolle Hilfe sein.

Frequenz	Note	Frequenz	Note
110	A	440	A (über mittlerem C)
117	A#,B♭	466	A#,B♭
123	H	494	H
131	C (tiefes)	523	C (hohes C)
139	C#,D♭	554	C#,D♭
147	D	587	D
156	D#,E♭	622	D#,E♭
165	E	659	E
175	F	698	F
185	F#,G♭	740	F#,G♭
196	G	784	G
208	G#,A♭	831	G#,A♭
220	A (unter mittlerem C)	880	A (über hohem C)
220	A (unter mittlerem C)	880	A (über hohem C)
233	A#,B♭	932	A#,B♭
247	H	988	H
262	C (mittleres C)	1047	C
277	C#,D♭	1109	C#,D♭
294	D	1175	D
311	D#,E♭	1245	D#,E♭
330	E	1319	E
349	F	1397	F
370	F#,G♭	1480	F#,G♭
392	G	1568	G
415	G#,A♭	1661	G#,A♭
440	A (über mittlerem C)	1760	A

SOUND-Unterprogramm

Bei Angabe eines negativen Wertes für die Frequenz wird kein Ton, sondern ein Rauschen erzeugt. Dabei handelt es sich entweder um ein "weißes Rauschen" oder ein "periodisches Rauschen". Das für jeden Wert zugeordnete Rauschen wird in der untenstehenden Tabelle angegeben. Da es sehr schwierig ist, in diesem Zusammenhang die akustischen Unterschiede zu beschreiben, probieren Sie die verschiedenen Rauschformen selbst aus, um mit jedem Klang vertraut zu werden.

Rausch-Charakteristika

Frequenzwert	Eigenschaft
—1	periodisches Rauschen Typ 1
—2	periodisches Rauschen Typ 2
—3	periodisches Rauschen Typ 3
—4	periodisches Rauschen, das mit der Frequenz des dritten spezifizierten Tons variiert
—5	weißes Rauschen Typ 1
—6	weißes Rauschen Typ 2
—7	weißes Rauschen Typ 3
—8	weißes Rauschen, das mit der Frequenz des dritten spezifizierten Tons variiert

Ein Maximum von drei Tönen und einem Rauschen ist gleichzeitig aktivierbar. Für jeden angegebenen Ton oder für jedes Rauschen muß die Lautstärke unmittelbar folgen.

Beispiele:

```
>NEW

>100 FOR NOISE=-1 TO -8 STEP
 -1
>110 CALL SOUND(1000,NOISE,2)

>120 NEXT NOISE
>130 END
>RUN

 --alle 8 verschiedenen Geräusche
   werden erzeugt

 ** DONE **
```

```
>CALL SOUND(2000,-3,5)

 --ein einzelnes Geräusch
   erklingt
```

```
>CALL SOUND(2500,440,2,659,5,
 880,10,-6,15)

 -- 3 Töne und ein Geräusch
   erklingen
>DUR=2500
>VOL=2
>C=262
>E=330
>G=392
>CALL SOUND(DUR,C,VOL,E,VOL,G
 ,VOL)

  --erzeugt eine C-Dur Tonleiter
```

KEY-Unterprogramm

CALL KEY *(Tastaturmodus, Rückmeldevariable, Statusvariable)*

Das KEY-Unterprogramm erlaubt die direkte Übertragung eines Zeichens von der Tastatur in das Programm. Damit erübrigt sich ein INPUT-Statement und man spart Zeit bei der Übernahme der Daten von einer Einzeltaste in den Arbeitsspeicher. Da das durch Tastendruck repräsentierte Zeichen nicht auf dem Bildschirm erscheint, werden mit der Ausführung des CALL KEY-Statements die Informationen, die bereits auf dem Schirm angezeigt sind, nicht beeinflußt. Der Tastaturmodus gibt an, welche Funktion die Tastatur hat. Er ist ein numerischer Ausdruck, der bei der Auswertung einen Wert zwischen 0 und 5 annimmt;

0 = Konsolentastatur
1 = linke Seite der Konsolentastatur oder Fernbedienungseinheit 1
2 = rechte Seite der Konsolentastatur oder Fernbedienungseinheit 2
3, 4, 5 = spezielle Tastaturmodi

Ein Tastaturmodus von 0 hebt die in vorhergehenden Programmzeilen mit CALL KEY definierten Tastaturmodi auf. Der Computer verwendet die eingebaute Standard-Tastatur.

Tastaturmodus 1 und 2 werden für eine geteilte Tastatur-Abfrage verwendet, wenn Sie das Tastenfeld in zwei Zweifach-Tastaturen trennen wollen oder die Aktivierungsknöpfe der Fernbedienung als Eingabeeinheiten benutzen wollen.

Bei der Angabe von 3, 4 oder 5 als Tastaturmodus erhält man spezielle Versionen. Der gewählte Modus bestimmt die Zeichenkodes, die von bestimmten Tasten erzeugt werden.

Mit dem Tastaturmodus 3 wählen Sie die Standard TI-99/4A Tastatur. (Die meisten Programm-Module verwenden diesen Modus.) Groß- und Kleinbuchstaben werden als Großbuchstaben wiedergegeben. Die Funktionstasten (BACK, BEGIN, CLEAR, etc.) erzeugen die Kodes 1 bis 15. Steuerzeichen sind nicht verfügbar.

Mit dem Tastaturmodus 4 wird die Tastatur auf PASCAL eingestellt. Hier werden die Kodes für Groß- und Kleinbuchstaben vom Computer erzeugt, ferner die Funktionstastenkodes von 129 bis 143. Die Kodes für die Steuerzeichen sind 1 bis 31.

Mit dem Tastaturmodus 5 wird die BASIC-Tastatur gewählt. Der Computer erzeugt Zeichenkodes für Groß- und Kleinbuchstaben. Die Kodes für die Funktionstasten sind 1 bis 15, für die Steuerzeichen 128 bis 159 und 187.

Die Rückmeldevariable muß eine numerische Variable sein. Der Computer setzt bei „Rückmeldevariable" den durch die gedrückte Taste repräsentierten numerischen Zeichenkode ein. Ist der Tastaturmodus 0 (Konsolentastatur), sind die Zeichenkodes die normalen ASCII-Kodes von 0 bis 127.

KEY-Unterprogramm (Fortsetzung)

Bei Anwendung der geteilten Tastatur (Tastaturmodus 1 und/oder 2) liegen die Zeichenkodes zwischen 0 und 19.

Die Statusvariable ist eine numerische Variable, die als Indikator dient, um Sie über das Geschehen auf der Tastatur zu informieren. Nach Durchführung der CALL KEY Routine ordnet der Computer der Statusvariablen einen der folgenden Kodes zu:

- +1 = seit der letzten Durchführung der CALL KEY Routine wurde eine neue Taste gedrückt
- -1 = während der Durchführung von CALL KEY wurde die gleiche Taste gedrückt wie bei der vorangehenden Durchführung
- 0 = es wurde keine Taste gedrückt.

Sie können dann Ihrem Programm diesen Status-Indikator kontrollieren, um die nächste Aktion zu bestimmen (siehe Zeile 110 im Programm rechts). Zeile 110 ist ein Test, der Ihnen Zeit gibt, eine andere Taste zu finden und zu drücken, ehe der Computer mit dem nächsten Statement fortfährt.

Tastaturmodus = 1 Tastaturmodus = 2

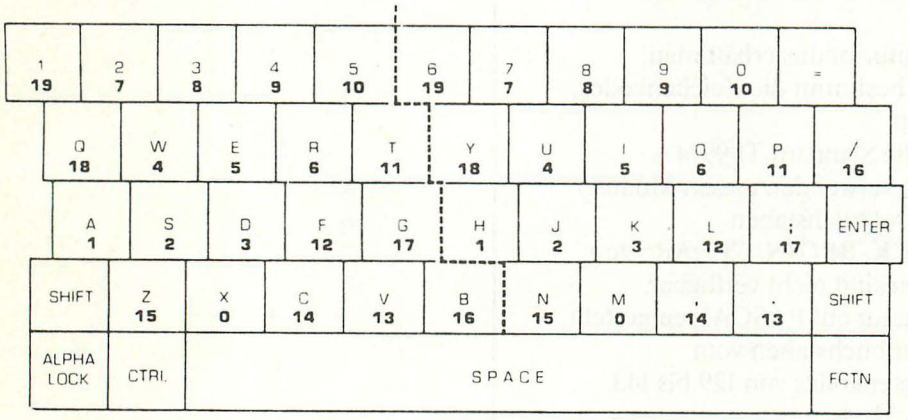


Abb. 1: Geteilte Tastatur
Funktionskodes 0 und 19

Tabelle 4: Zeichenkodes für die geteilte Tastatur

Kodes	Tasten*	Kodes	Tasten*	
0	X,M	10	5,0	
1	A,H	11	T,P	
2	S,J	12	F,L	
3	D,K	13	V, .	(Punkt)
4	W,U	14	C, ,	(Komma)
5	E,I	15	Z, N	
6	R,0	16	B, /	(Schrägstrich)
7	2,7	17	G, ;	(Semikolon)
8	3,8	18	Q,Y	
9	4,9	19	1,6	

Beispiele:

```
>NEW
>100 CALL KEY(0,KEY,STATUS)
>110 IF STATUS=0 THEN 100
>120 NOTE=KEY-64
>130 ON NOTE GOTO 250,270,150
 ,170,190,210,230
>140 GOTO 100
>150 NOTE=262
>160 GOTO 280
>170 NOTE=294
>180 GOTO 280
>190 NOTE=330
>200 GOTO 280
>210 NOTE=349
>220 GOTO 280
>230 NOTE=392
>240 GOTO 280
>250 NOTE=440
>260 GOTO 280
>270 NOTE=494
>280 CALL SOUND(100,NOTE,2)
>290 GOTO 100
>RUN
```

-- (Beim Drücken einer Taste (A-G) erklingt der entsprechende Ton der Tonleiter)

(Durch CLEAR wird das Programm abgebrochen)

KEY-Unterprogramm (Fortsetzung)

3 1	4 2	7 3	2 4	14 5	12 6	1 7	6 8	15 9	0	5 =
Q	W	11 E	R	T	Y	U	I	O	P	/
A	8 S	9 D	F	G	H	J	K	L	;	13 ENTER
SHIFT	Z	10 X	C	V	B	N	M	,	.	SHIFT
ALPHA LOCK	CTRL	SPACE								FCTN

Abb. 2: Standard TI-99/4A Tastatur
Tastaturmodus = 3. Groß- und Kleinbuchstaben werden als Großbuchstaben wiedergegeben. Funktionskodes von 1 bis 15. Steuerzeichenkodes können nicht erzeugt werden.

131 1	132 2	135 3	130 4	142 5	140 6	129 7	134 8 30	143 9 31	0	133 = 29
Q 17	W 23	139 E 5	R 18	T 20	Y 25	U 21	I 9	O 15	P 16	/
A 1	136 S 19	137 D 4	F 6	G 7	H 8	J 10	K 11	L 12	; 28	141 ENTER
SHIFT	Z 26	138 X 24	C 3	V 22	B 2	N 14	M 13	,	. 27	SHIFT
ALPHA LOCK	CTRL	SPACE								FCTN

Abb. 3: PASCAL-Tastatur
Tastaturmodus = 4. Groß- und Kleinbuchstaben möglich. Funktionskodes 129 bis 143. Steuerzeichenkodes 1 bis 31.

3 1	4 2	7 3	2 4	14 5	12 6	1 7	6 8 158	15 9 159	0	5 = 157
Q 145	W 151	11 E 133	R 146	T 148	Y 153	U 149	I 137	O 143	P 144	/ 187
A 129	8 S 147	9 D 132	F 134	G 135	H 136	J 138	K 139	L 140	; 156	13 ENTER
SHIFT	Z 154	10 X 152	C 131	V 150	B 130	N 142	M 141	, 128	. 155	SHIFT
ALPHA LOCK	CTRL	SPACE								FCTN

Abb. 4: BASIC-Tastatur
Tastaturmodus = 5. Groß- und Kleinbuchstaben möglich. Funktionskodes 1 bis 15. Steuerzeichenkodes 128 bis 159, 187.

Kodes für Funktions- und Steuertasten

Kodes für Funktions- und Steuertasten
Den Funktions- und Steuertasten sind ebenfalls Kodes zugeordnet, so daß sie über das CALL KEY Unterprogramm im TI-BASIC angesprochen werden können. Die zugeordneten Kodes hängen vom Tastaturmodus ab, der in einem CALL KEY Programm-Statement spezifiziert wurde.

Kodes für Funktionstasten

TI-99/4A BASIC-Modi	Pascal-Modus	Funktionsbezeichnung	Funktions-Taste
1	129	**AID**	**FCTN 7**
2	130	**CLEAR**	**FCTN 4**
3	131	**DEL**ete	**FCTN 1**
4	132	**INS**ert	**FCTN 2**
5	133	**QUIT**	**FCTN =**
6	134	**REDO**	**FCTN 8**
7	135	**ERASE**	**FCTN 3**
8	136	**LEFT** (Pfeil links)	**FCTN S**
9	137	**RIGHT** (Pfeil rechts)	**FCTN D**
10	138	**DOWN** (Pfeil unten)	**FCTN X**
11	139	**UP** (Pfeil oben)	**FCTN E**
12	140	**PROD'D**	**FCTN 6**
13	141	**ENTER**	**ENTER**
14	142	**BEGIN**	**FCTN 5**
15	143	**BACK**	**FCTN 9**

Kodes für Steuertasten

BASIC Modus	Pascal-Modus	Kurzzeichen	Taste	ANMERKUNGEN
129	1	SOH	CONTROL A	Anfang des Kopfes
130	2	STX	CONTROL B	Anfang des Textes
131	3	ETX	CONTROL C	Ende des Textes
132	4	EOT	CONTROL D	Ende der Übertragung
133	5	ENQ	CONTROL E	Stationsaufforderung
134	6	ACK	CONTROL F	Positive Rückmeldung
135	7	BEL	CONTROL G	Klingel
136	8	BS	CONTROL H	Rückwärtsschritt
137	9	HT	CONTROL I	Horizontal-Tabulator
138	10	LF	CONTROL J	Zeilenvorschub
139	11	VT	CONTROL K	Vertikal-Tabulator
140	12	FF	CONTROL L	Formularvorschub
141	13	CR	CONTROL M	Wagenrücklauf
142	14	SO	CONTROL N	Dauerumschaltung
143	15	SI	CONTROL O	Rückschaltung
144	16	DLE	CONTROL P	Datenübertragungsumschaltung
145	17	DC1	CONTROL Q	Greätesteuerzeichen (X-EIN)
146	18	DC2	CONTROL R	Gerätesteuerzeichen
147	19	DC3	CONTROL S	Gerätesteuerzeichen (X-AUS)
148	20	DC4	CONTROL T	Gerätesteuerzeichen
149	21	NAK	CONTROL U	Negative Rückmeldung
150	22	SYN	CONTROL V	Synchronisierung
151	23	ETB	CONTROL W	Ende des Datenübertragungsblocks
152	24	CAN	CONTROL X	Ungültig
153	25	EM	CONTROL Y	Ende der Aufzeichnung
154	26	SUB	CONTROL Z	Substitionszeichen
155	27	ESC	CONTROL .	Code-Umschaltung
156	28	FS	CONTROL ;	Hauptgruppen-Trennzeichen
157	29	GS	CONTROL =	Gruppen-Trennzeichen
158	30	RS	CONTROL 8	Untergruppen-Trennzeichen
159	31	US	CONTROL 9	Teilgruppen-Kennzeichen

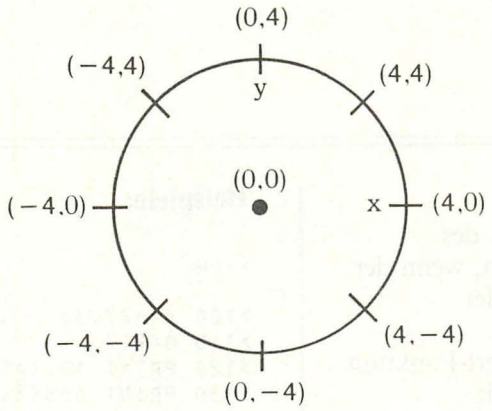
JOYST-Unterprogramm

CALL JOYST *(Tastaturmodus, x-Wert, y-Wert)*

Das JOYST-Unterprogramm erlaubt die Eingabe von Informationen an den Computer, basierend auf der Position des Hebels der Fernbedienung (Zubehör).
Die Tasteneinheit ist ein numerischer Ausdruck, der nach der Auswertung einen Wert zwischen 1 und 4 haben muß.

- 1 = Kontroller 1
- 2 = Kontroller 2
- 3, 4 und 5 = spezielle Tastaturmodi

Bei der Angabe von 3, 4 oder 5 als Tastaturmodis erhält man spezielle Versionen, die im KEY-Unterprogramm eingehend erklärt sind. Wenn für den Tastaturmodus der Wert 3, 4 oder 5 angegeben wurde, kann der Computer Eingaben von der Fernbedienung nicht richtig identifizieren.
Für x- und y-Wert müssen numerische Variable verwendet werden. Der Computer ordnet diesen Variablen einen ganzzahligen Wert von −4, +4 oder 0 zu, basierend auf der Position des "Steuerknüppels" zu diesem Zeitpunkt (siehe Abbildung unten). Die erste Zahl in Klammern ist der x-Wert, die zweite der y-Wert.



Sie können dann diese Werte in Ihrem Programm durch Bezugnahme auf die Variablenbezeichnungen verwenden.
Detaillierte Instruktionen finden Sie in der Bedienungsanleitung der Fernbedienung.

```
>NEW

>100 CALL CLEAR
>110 CALL CHAR(42,"FFFFFFFFFF
 FFFFFF")
>120 INPUT "SCREEN COLOR?":S
>130 INPUT "BLOCK COLOR?":F
>140 CALL CLEAR
>150 CALL SCREEN(S)
>160 CALL COLOR(2,F,1)
>170 CALL JOYST(2,X,Y)
>180 A=X*2.2+16.6
>190 B=Y*1.6+12.2
>200 CALL HCHAR(B,A,42)
>210 GOTO 170
>RUN

 --Bildschirm wird gelöscht

 SCREEN COLOR?14
 BLOCK COLOR?9

 --Bildschirm wird gelöscht

--Farbblock wird auf Bildschirm
  bewegt, entsprechend der
  Auslenkung der Fernbedienung

  (Durch CLEAR wird das
  Programm abgebrochen)
```

Eingebaute numerische Funktionen

Einführung

Viele Spezialfunktionen sind beim TI Home Computer System bereits einprogrammiert:

- arithmetische Funktionen
- String-Funktionen
- anwenderdefinierbare Funktionen
- Datenfelder
- Unterprogramme
- Dateiverwaltung
  (jeweils in alphabetischer Reihenfolge)

Extended Basic erweitert die eingebauten numerischen Funktionen noch um viele Operationen, die dem Anwender umständliche Routineprogrammierungen abnehmen (z. B. MAX; MIN; PI; REC; RPT$; SEG$; u. a.)*

ABS - Absolutwert

ABS *(numerischer Ausdruck)*

Die Absolutwert-Funktion gibt Ihnen den absoluten Wert des Arguments an. Das Argument ist der Wert, den Sie erhalten, wenn der numerische Ausdruck berechnet wird. Für die Berechnung der numerischen Ausdrücke finden die üblichen Regeln Anwendung. Ist das Argument positiv, ergibt die Absolutwert-Funktion das Argument selbst. Bei negativem Argument gibt Ihnen die Absolutwert-Funktion den Negativwert des Arguments. Für ein Argument X gilt also:

- Wenn $X \geqslant 0$, ABS (X) = X
- Wenn $X < 0$, ABS (X) = —X (d.h., ABS (—3) = — (—3) = 3)

Beispiele:

```
>NEW

>100 A=-27.36
>110 B=9.7
>120 PRINT ABS(A);ABS(B)
>130 PRINT ABS(3.8);ABS(-4.5)

>140 PRINT ABS(-3*2)
>150 PRINT ABS(A*(B-3.2))
>160 END
>RUN
  27.36  9.7
  3.8  4.5
  6
  177.84

 ** DONE **
```

* separat als Zubehör

ATN - Arcustangens

ATN *(numerischer Ausdruck)*

Mit der Arcustangens-Funktion erhalten Sie den Arcustangens des Arguments. Das Argument ist der Wert, den Sie erhalten, wenn der numerische Ausdruck berechnet wird. Für die Berechnung der numerischen Ausdrücke finden die üblichen Regeln Anwendung. Also ergibt ATN (x) den Winkel (in Bogenmaß), dessen Tangens x ist. Wenn Sie den äquivalenten Winkel in Grad haben wollen, müssen Sie das Resultat mit (180/(4 ATN(1))) oder mit 57.295779513079 (dies entspricht 180/ π) multiplizieren. Der Wert aus der Arcustangens-Funktion liegt immer im Bereich $-\pi/2 < ATN(X) < \pi/2$.

Beispiele:

```
>NEW

>100 PRINT ATN(.44)
>110 PRINT ATN(1E127)
>120 PRINT ATN(1E-129);ATN(0)

>130 PRINT ATN(.3)*57.2957795
 13079
>140 PRINT ATN(.3)*(180/(4*AT
 N(1)))
>150 END
>RUN
  .4145068746
  1.570796327
  0  0
  16.69924423
  16.69924423

 ** DONE **
```

COS - Kosinus

COS *(numerischer Ausdruck)*

Die Kosinus-Funktion gibt Ihnen den Kosinus des Arguments x an, wobei x ein Winkel in Bogenmaß ist. Das Argument ist der Wert, den Sie erhalten, wenn der numerische Ausdruck berechnet wird. Für die Berechnung von numerischen Ausdrücken werden die üblichen Regeln angewendet. Ist Ihr Winkel in Grad ausgedrückt, multiplizieren Sie diesen Wert mit π /180, um den äquivalenten Winkel in Bogenmaß zu erhalten. Für π /180 können Sie (4*ATN(1))/180 oder 0.01745329251994 einsetzen. Achtung: Wenn Sie einen x-Wert eingeben, bei dem $|x| \geqslant 1.5707963266375*10^{10}$ ist, wird die Nachricht "BAD ARGUMENT" (falsches Argument) angezeigt und das Programm gestoppt.

Beispiele:

```
>NEW

>100 A=1.047197551196
>110 B=60
>120 C=.01745329251994
>130 PRINT COS(A);COS(B*C)
>140 PRINT COS(B*(4*ATN(1))/1
 80)
>150 END
>RUN
  .5  .5
  .5

 ** DONE **

>PRINT COS(2.2E11)

 * BAD ARGUMENT
```

EXP - Exponentialfunktion

EXP *(numerischer Ausdruck)*

Die Exponentialfunktion gibt Ihnen den Wert von e^x an, wobei e = 2.718281828 ist. Das Argument x ist der Wert, den Sie erhalten, wenn der numerische Ausdruck berechnet wird. Für die Berechnung von numerischen Ausdrücken finden die üblichen Regeln Anwendung. Die Exponentialfunktion ist die Umkehrung für den natürlichen Logarithmus (Funktion LOG. Daher gilt: X = EXP(LOG(X)).

```
>NEW

>100 A=3.79
>110 PRINT EXP(A);EXP(9)
>120 PRINT EXP(A*2)
>130 PRINT EXP(LOG(2))
>140 END
>RUN
  44.25640028  8103.083928
  1958.628965
  2

 ** DONE **
```

INT - Ganzzahlfunktion

INT *(numerischer Ausdruck)*

In der Ganzzahlfunktion erhalten Sie die größte ganze Zahl, die nicht größer ist als das Argument. Das Argument ist der Wert, den Sie erhalten, wenn der numerische Ausdruck berechnet wird. Für die Berechnung von numerischen Ausdrücken werden die üblichen Regeln angewendet. Die Ganzzahlfunktion gibt Ihnen immer die nächste ganze Zahl an, die sich links von der spezifizierten Zahl auf der Zeile befindet. D.h., für positive Zahlen wird der Dezimalteil weggelassen; für negative Zahlen wird der nächstkleinere ganzzahlige Wert verwendet (z.B., INT(—2.3) = —3). Bei Angabe einer ganzen Zahl bleibt mit dieser Funktion die gleiche ganze Zahl erhalten.

Beispiele:

```
>NEW

>100 B=.678
>110 A=INT(B*100+.5)/100
>120 PRINT A;INT(B)
>130 PRINT INT(-2.3);INT(2.2)

>140 STOP
>RUN
  .68  0
 -3  2

 ** DONE **
```

LOG - Natürlicher Logarithmus

LOG *(numerischer Ausdruck)*

Die Funktion für den natürlichen Logarithmus gibt Ihnen den natürlichen Logarithmus der durch das Argument spezifizierten Zahl an. Das Argument ist der Wert, den Sie erhalten, wenn der numerische Ausdruck berechnet wird. Für die Berechnung von numerischen Ausdrücken werden die üblichen Regeln angewendet. Die Schreibweise für den natürlichen Logarithmus von x ist im allgemeinen $\log_e$ (x). Die Logarithmusfunktion ist die Umkehrung der Exponentialfunktion EXP. Daher gilt: X = LOG(EXP(X)).

Das Argument für die Funktion des natürlichen Logarithmus muß größer als Null sein. Wenn Sie für das Argument einen Wert kleiner oder gleich 0 angeben, wird die Nachricht "BAD ARGUMENT" (falsches Argument) angezeigt und der Programmablauf unterbrochen.

Zur Berechnung des Logarithmus einer Zahl mit einer anderen Basis, B, verwenden Sie folgende Formel:

$\log_B (x) = \log_e (x)/\log_e (B)$

Zum Beispiel:

$\log_{10} (3) = \log_e (3)/\log_e (10)$

```
>NEW

>100 A=3.5
>110 PRINT LOG(A);LOG(A*2)
>120 PRINT LOG(EXP(2))
>130 STOP
>RUN
  1.252762968  1.945910149
  2.

 ** DONE **


>PRINT LOG(-3)

 * BAD ARGUMENT


>PRINT LOG(3)/LOG(10)
  .4771212547
```

RANDOMIZE - Statement (Zufallszahlen)

RANDOMIZE [*Anfangszahl*]

Das RANDOMIZE-Statement wird in Verbindung mit der Zufallszahlenfunktion (RND) verwendet. Ohne Anwendung des RANDOMIZE-Statements erzeugt die Zufallszahlenfunktion bei jedem Programmlauf immer die gleiche Folge von Pseudo-Zufallszahlen. Verwendet man das RANDOMIZE-Statement ohne Anfangszahl, erzeugt die Zufallszahlenfunktion bei jedem Programmlauf eine andere und unvorhersagbare Folge von Zufallszahlen.

Wenn Sie mit dem RANDOMIZE-Statement eine Anfangszahl spezifizieren, ist die Folge der Zufallszahlen, die mit der Zufallszahlenfunktion erzeugt werden, vom Wert dieser Anfangszahl abhängig. Bei Anwendung der gleichen Anfangszahl in jedem Programmlauf wird auch die gleiche Zahlenfolge erzeugt. Umgekehrt erhalten Sie eine andere Zahlenfolge, wenn bei jedem Programmlauf eine andere Anfangszahl eingesetzt wird. Die Anfangszahl kann ein beliebiger numerischer Ausdruck sein. Die tatsächlich verwendete Zahl besteht aus den ersten beiden Bytes der internen Darstellung dieser Zahl. Es ist also möglich, daß die gleiche Zahlenfolge erzeugt wird, selbst wenn Sie unterschiedliche Anfangszahlen spezifizieren. RANDOMIZE 1000 und RANDOMIZE 1099 ergeben zum Beispiel intern zwei identische erste Bytes, und erzeugen somit auch die gleiche Zahlenfolge. Ist die angegebene Anfangszahl eine Dezimalzahl so wird der ganzzahlige Anfang (Integer) verwendet.
(siehe hierzu die Funktion INT).

Beispiele:

```
>NEW

>100 RANDOMIZE 23
>110 FOR I=1 TO 5
>120 PRINT INT(10*RND)+1
>130 NEXT I
>140 STOP
>RUN
  6
  4
  3
  8
  8

 ** DONE **
```

RND - Zufallszahlen

RND

Die Zufallszahlenfunktion gibt Ihnen die nächste Pseudo-Zufallszahl in der momentanen Folge der Pseudo-Zufallszahlen an. Die erzeugte Zufallszahl ist größer oder gleich null und kleiner als 1. Die Zahlenfolge, die mit der Zufallszahlenfunktion erzeugt wird, ist bei jedem Programmlauf gleich, wenn nicht ein RANDOMIZE-Statement im Programm erscheint.

```
>NEW
>100 FOR I=1 TO 5
>110 PRINT INT(10*RND)+1
>120 NEXT I
>130 END
>RUN
  6
  4
  6
  4
  3
 ** DONE **
```

Wenn Sie ganze Zufallszahlen zwischen zwei Werten A und B (A < B) inklusive, erhalten wollen, wenden Sie die folgende Formel an:
INT((B—A + 1)*RND) + A

```
>NEW
>100 REM RANDOM INTEGERS
 BETWEEN 1 AND 20,INCLUSIVE
>110 FOR I=1 TO 5
>120 C=INT(20*RND)+1
>130 PRINT C
>140 NEXT I
>150 END
>RUN
  11
  8
  11
  8
  6
 ** DONE **
```

SGN - Signum (Vorzeichen)

SGN *(numerischer Ausdruck)*

Die Signum-Funktion gibt Ihnen das algebraische Vorzeichen des Wertes an, der durch das Argument gegeben ist. Das Argument ist der Wert, den Sie erhalten, wenn der numerische Ausdruck berechnet wird. Für die Berechnung von numerischen Ausdrücken werden die üblichen Regeln angewendet. Für die Signum-Funktion ergeben sich verschiedene Werte, abhängig vom Wert des Arguments. Diese Werte sind nachstehend aufgelistet. Für das Argument X gilt:

- $X < 0$, SGN (X) = —1
- $X = 0$, SGN (X) = 0
- $X > 0$, SGN (X) = 1

Beispiele:

```
>NEW

>100 A=-23.7
>110 B=6
>120 PRINT SGN(A);SGN(0);SGN(
 B)
>130 PRINT SGN(-3*3);SGN(B*2)

>140 END
>RUN
 -1  0  1
 -1  1

 ** DONE **
```

SIN - Sinus

SIN *(numerischer Ausdruck)*

Mit der Sinus-Funktion erhalten Sie den Sinus für das Argument x, wobei x ein Winkel in Bogenmaß ist. Das Argument ist der Wert, den Sie erhalten, wenn der numerische Ausdruck berechnet wird. Für die Berechnung von numerischen Ausdrücken werden die üblichen Regeln angewendet. Wenn der Winkel in der Einheit Grad angegeben ist, multiplizieren Sie die Gradzahl einfach mit $\pi/180$, um den äquivalenten Winkel in Bogenmaß zu erhalten. Für $\pi/180$ können Sie (4*ATN(1))/180 oder 0.01745329251944 einsetzen.

Achtung: Bei Eingabe eines Wertes von x, wo $|x| \geqslant 1.5707963266375 * 10^{10}$ ist, wird die Nachricht "BAD ARGUMENT" (falsches Argument) angezeigt und der Programmablauf unterbrochen.

```
>NEW

>100 A=.5235987755982
>110 B=30
>120 C=.01745329251994
>130 PRINT SIN(A);SIN(B*C)
>140 PRINT SIN(B*(4*ATN(1))/1
 80)
>150 END
>RUN
  .5   .5
  .5

 ** DONE **


>PRINT SIN(1.9E12)

 * BAD ARGUMENT
```

SQR - Quadratwurzelfunktion

SQR *(numerischer Ausdruck)*

Die Quadratwurzelfunktion gibt Ihnen die positive Quadratwurzel des Wertes an, der durch das Argument gegeben ist. Das Argument ist der Wert, den Sie erhalten, wenn der numerische Ausdruck berechnet wird. Für die Berechnung von numerischen Ausdrücken werden die üblichen Regeln angewendet. SQR(x) ist die Entsprechung zu x/\(1/2). Der durch das Argument gegebene Wert darf nicht negativ sein.

Wenn der Wert für das Argument kleiner als Null ist, wird die Nachricht "BAD ARGUMENT" (falsches Argument) angezeigt und der Programmablauf unterbrochen.

Beispiele:

```
>NEW

>100 PRINT SQR(4);4^(1/2)
>110 PRINT SQR(10)
>120 END
>RUN
  2  2
  3.16227766

 ** DONE **

>PRINT SQR(-5)

 * BAD ARGUMENT
```

TAN - Tangens

TAN *(numerischer Ausdruck)*

Die Tangensfunktion gibt Ihnen den Tangens des Arguments x an, wobei x ein Winkel in Bogenmaß ist. Das Argument ist der Wert, den Sie erhalten, wenn der numerische Ausdruck berechnet wird. Für die Berechnung von numerischen Ausdrücken werden die üblichen Regeln angewendet. Wenn der Winkel in der Einheit Grad ausgedrückt ist, multiplizieren Sie ihn mit $\pi/180$, um den äquivalenten Winkel in Bogenmaß zu erhalten. Für $\pi/180$ können Sie (4*ATN(1))/180 oder 0.01745329251994 einsetzen.

Achtung: Wenn Sie einen Wert von x eingeben, wo $|x| \geq 1.5707963266375 * 10^{10}$ ist, wird die Nachricht "BAD ARGUMENT" angezeigt und das Programm unterbrochen.

```
>NEW

>100 A=.7853981633973
>110 B=45
>120 C=.01745329251994
>130 PRINT TAN(A);TAN(B*C)
>140 PRINT TAN(B*(4*ATN(1))/1
 80)
>150 END
>RUN
  1.  1.
  1

 ** DONE **

>PRINT TAN(1.76E10)

 * BAD ARGUMENT
```

Eingebaute String-Funktionen

Einführung

Neben den eingebauten arithmetischen Funktionen sind noch viele weitere in TI-BASIC festprogrammiert. Die in diesem Abschnitt behandelten Funktionen werden als String-Funktionen bezeichnet. String-Funktionen verwenden entweder einen String in bestimmter Weise, um ein numerisches Ergebnis zu erzeugen, oder das Ergebnis der Auswertung der Funktion ist ein String. Mit dem Gebrauch des Computers werden Sie viele Möglichkeiten finden, die hier beschriebenen Funktionen zu nutzen. Sie können auch Ihre eigenen String-Funktionen definieren (siehe hierzu benutzerdefinierte Funktionen).

Beachten Sie, daß eine String-Funktion mit einem Namen, der mit einem Dollarzeichen endet (z. B. CHR$), immer wieder nur ein String-Resultat ergibt, und nicht in numerischen Ausdrücken verwendet werden kann.

ASC – ASCII-Wert

ASC *(String-Ausdruck)*

Die ASCII-Wert-Funktion gibt Ihnen den ASCII-Zeichenkode an, der dem ersten Zeichen des durch den String-Ausdruck spezifizierten Strings entspricht. Eine Liste der ASCII-Zeichenkodes für jedes Zeichen im Standard-Zeichensatz ist im Anhang angegeben.

Beispiele:

```
>NEW

>100 A$="HELLO"
>110 C$="JACK SPRAT"
>120 C=ASC(C$)
>130 B$="THE ASCII VALUE OF "

>140 PRINT B$;"H IS";ASC(A$)
>150 PRINT B$;"J IS";C
>160 PRINT B$;"N IS";ASC("NAM
 E")
>170 PRINT B$;"1 IS";ASC("1")

>180 PRINT CHR$(ASC(A$))
>190 END
>RUN
  THE ASCII VALUE OF H IS 72
  THE ASCII VALUE OF J IS 74
  THE ASCII VALUE OF N IS 78
  THE ASCII VALUE OF 1 IS 49
  H

 ** DONE **
```

ASCII-Zeichenkodes

Tabelle 1: Zeichenkodes

ASCII-Zeichencodes

Die für den TI-99/4A Computer definierten Zeichen sind die Standard ASCII-Zeichen für die Kodes von 32 bis 127. In der folgenden Tabelle sind die Zeichen und ihre Codes aufgelistet.

ASCII KODE	ZEICHEN		ASCII KODE	ZEICHEN		ASCII KODE	Zeichen	
32		(Leerzeichen)	64	@	("at"-Zeichen)	96	'	(Apostroph)
33	!	(Ausrufezeichen)	65	A		97	a	
34	"	(Anführungszeichen)	66	B		98	b	
35	#	(Nummer- oder Pfund-Zeichen)	67	C		99	c	
36	$	(Dollar)	68	D		100	d	
37	%	(Prozent)	69	E		101	e	
38	&	(Und-Zeichen)	70	F		102	f	
39	'	(Apostroph)	71	G		103	g	
						104	h	
40	(	(offene Klammer)	72	H		105	i	
41	)	(geschl. Klammer)	73	I		106	j	
42	*	(Sternchen)	74	J		107	k	
43	+	(plus)	75	K		108	l	
44	,	(Komma)	76	L		109	m	
45	-	(minus)	77	M		110	n	
46	.	(Punkt)	78	N		111	o	
47	/	(Schrägstrich)	79	O		112	p	
						113	q	
48	0		80	P		114	r	
49	1		81	Q		115	s	
50	2		82	R		116	t	
51	3		83	S		117	u	
52	4		84	T		118	v	
53	5		85	U		119	w	
54	6		86	V		120	x	
55	7		87	W		121	y	
						122	z	
56	8		88	X		123	{	(linke geschweifte Klammer)
57	9		89	Y		124	¦	(Doppelstrich)
58	:	(Doppelpunkt)	90	Z		125	}	(rechte geschweifte Klammer)
59	;	(Semikolon)	91	[	(offene eckige Klammer)	126	~	(Tilde)
60	<	(kleiner als)	92	\	(umgek. Schrägstrich)	127	DEL	(erscheint auf dem Bildschirm als Leerzeichen).
61	=	(gleich)	93	]	(geschl. eckige Klammer)			
62	>	(größer als)	94	∧	(Potenzierung)			
63	?	(Fragezeichen)	95	_	(Zeile)			

Zur Anwendung in Farbgrafik-Programmen sind diese Zeichenkodes in 12 Zeichensätze eingeteilt

Zeichensatz Nr.	Zeichen-Kodes	Zeichensatz Nr.	Zeichen-Kodes	Zeichensatz Nr.	Zeichen-Kodes
1	32-39	5	64-71	9	69-103
2	40-47	6	72-79	10	104-111
3	48-55	7	80-87	11	112-119
4	56-63	8	88-95	12	120-127

Zwei weitere Zeichen sind auf dem TI-99/4A Computer definiert. Dem Positionsanzeiger (Cursor) hat den ASCII-Kode 30, und das Randzeichen den Kode 31.

CHR$ – Zeichen

CHR$ *(numerischer Ausdruck)*

Mit der Zeichenfunktion erhalten Sie das Zeichen, das dem im Argument spezifizierten ASCII-Zeichenkode entspricht. Das Argument ist der Wert, den Sie erhalten, wenn der numerische Ausdruck berechnet wird. Für die Berechnung von numerischen Ausdrücken werden die üblichen Regeln angewendet. Ist das angegebene Argument keine ganze Zahl, wird es auf eine ganze Zahl gerundet.

Ist das angegebene Argument ein Wert zwischen 32 und 127 inklusive, dann erhält man ein Standardzeichen. Wenn das angegebene Argument zwischen 128 und 159 inklusive liegt, und für diesen Wert ein spezielles Graphikzeichen definiert wurde, dann wird das Graphikzeichen angegeben. Wenn Sie ein Argument spezifizieren, das ein undefiniertes Zeichen benennt (kein Standardzeichen oder kein definiertes Graphikzeichen), dann entspricht das Zeichen, das Sie erhalten, dem jeweiligen Speicherinhalt.

Wenn Sie für das Argument einen Wert angeben, der kleiner als 0 oder größer als 32767 ist, wird die Nachricht "BAD VALUE" angezeigt und das Programm unterbrochen.

Beispiele:

```
>NEW

>100 A$=CHR$(72)&CHR$(73)&CHR
 $(33)
>110 PRINT A$
>120 CALL CHAR(97,"0103070F1F
 3F7FFF")
>130 PRINT CHR$(32);CHR$(97)
>140 PRINT CHR$(3*14)
>150 PRINT CHR$(ASC("+"))
>160 END
>RUN
 HI!

 *
 +

 ** DONE **



>PRINT CHR$(33010)

 * BAD VALUE
```

LEN - Länge

LEN *(String-Ausdruck)*

Die Längenfunktion gibt Ihnen die Anzahl der Zeichen im String an, der durch das Argument gegeben ist. Das Argument ist der String-Wert, den Sie erhalten, wenn der String-Ausdruck ausgewertet wird. Zur Auswertung der String-Ausdrücke werden die üblichen Regeln angewandt.

```
>NEW

>100 NAME$="CATHY"
>110 CITY$="NEW YORK"
>120 MSG$="HELLO "&"THERE!"
>130 PRINT NAME$;LEN(NAME$)
>140 PRINT CITY$;LEN(CITY$)
>150 PRINT MSG$;LEN(MSG$)
>160 PRINT LEN(NAME$&CITY$)
>170 PRINT LEN("HI!")
>180 STOP
>RUN
 CATHY  5
 NEW YORK  8
 HELLO THERE!  12
  13
  3

 ** DONE **
```

POS – Position

POS *(String 1, String 2, numerischer Ausdruck)*

Die Positionsfunktion findet die erste Stelle von String 2 innerhalb von String 1. Sowohl String 1 als auch String 2 sind String-Ausdrücke. Der numerische Ausdruck wird ausgewertet, und, wenn nötig, auf eine ganze Zahl n gerundet. Für die Auswertung von String-Ausdrücken (Seite 35) und von numerischen Ausdrücken (Seite 34) werden die üblichen Regeln angewendet. Die Suche nach String 2 beginnt bei der n-ten Stelle von String 1. Wenn String 2 aufgefunden ist, wird die Position des ersten Zeichens von String 2 angegeben, der sich innerhalb von String 1 befindet. Wenn String 2 nicht aufgefunden wird, erhalten Sie einen Wert von null. Die Position des ersten Zeichens von String 1 ist die Stelle 1. Wenn Sie für n einen Wert angeben, der größer ist als die Anzahl der Zeichen in String 1, erhalten Sie ebenfalls einen Wert von null. Ist der für n spezifizierte Wert kleiner als null, wird die Nachricht "BAD VALUE" (falscher Wert) angezeigt, und der Programmablauf unterbrochen.

Beispiele:

```
>NEW

>100 MSG$="HELLO THERE! HOW A
 RE YOU?"
>110 PRINT "H";POS(MSG$,"H",1
 )
>120 C$="RE"
>130 PRINT C$;POS(MSG$,C$,1);
 POS(MSG$,C$,12)
>140 PRINT "HI";POS(MSG$,"HI"
 ,1)
>150 END
>RUN
 H  1
 RE  10  19
 HI  0

 ** DONE **
```

SEG$ - String-Segment

SEG$ *(String-Ausdruck, numerischer Ausdruck 1, numerischer Ausdruck 2)*

Mit der Stringsegment-Funktion erhalten Sie einen Teil des durch den String-Ausdruck bezeichneten Strings (Sub-String). Der numerische Ausdruck 1 identifiziert die Position des Zeichens im Originalstring, das gleichzeitig das erste Zeichen im Sub-String darstellt.

Das erste Zeichen im angegebenen String ist in der Position 1. Die Länge des Sub-Strings wird durch den numerischen Ausdruck 2 spezifiziert. In der Auswertung der numerischen Ausdrücke und der String-Ausdrücke werden die üblichen Regeln angewendet.

Beispiele:

```
>NEW

>100 MSG$="HELLO THERE! HOW A
 RE YOU?"
>110 REM SUBSTRING BEGINS IN
 POSITION 14 AND HAS A LENGTH
  OF 12.
>120 PRINT SEG$(MSG$,14,12)
>130 END
>RUN
 HOW ARE YOU?

 ** DONE **
```

Für diese Erläuterung wird A$ für den String-Ausdruck, X für den numerischen Ausdruck 1 und Y für den numerischen Ausdruck 2 verwendet.

Wenn Sie für X einen Wert größer als die Länge von A$ (Zeile 110) oder für Y einen Wert von 0 (Zeile 120) angeben, dann erhalten Sie den Null-String. Wenn Sie für Y einen Wert größer als die restliche Länge in A$ angeben, ausgehend von der durch X festgelegten Position (Zeile 130), erhalten Sie den Rest von A$ ab dieser Position.

```
>NEW

>100 MSG$="I AM A COMPUTER."
>110 PRINT SEG$(MSG$,20,1)
>120 PRINT SEG$(MSG$,10,0)
>130 PRINT SEG$(MSG$,8,20)
>140 END
>RUN

 COMPUTER.

 ** DONE **
```

Bei Angabe eines X-Wertes kleiner oder gleich null und/oder eines Y-Wertes kleiner null wird die Nachricht "BAD VALUE" (falscher Wert) angezeigt und der Programmablauf unterbrochen.

```
>PRINT SEG$(MSG$,-1,10)

 * BAD VALUE
```

STR$ - Stringzahl

STR$ *(numerischer Ausdruck)*

Die Stringzahl-Funktion formt die durch das Argument spezifizierte Zahl in einen String um. Das Argument ist der Wert, den Sie erhalten, wenn der numerische Ausdruck berechnet wird. Für die Berechnung von numerischen Ausdrücken werden die üblichen Regeln angewendet. Wird die Zahl in einen String umgeformt, ist der String eine gültige Darstellung einer numerischen Konstanten ohne vorangestellte oder nachfolgende Leerzeichen. Zum Beispiel: wenn B = 69.5, dann ist STR$ (B) der String "69.5". Nur String-Operationen können mit den Strings durchgeführt werden, die mit einer Stringzahl-Funktion gebildet wurden. Die Stringzahl-Funktion ist die Umkehrung der Value-Funktion (VAL), die unten beschrieben ist. Beachten Sie in dem Beispiel, daß bei den in Strings umgeformten Zahlen keine vorangestellten nachfolgenden Leerzeichen vorhanden sind.

Beispiele:

```
>NEW

>100 A=-26.3
>110 PRINT STR$(A);" ";A
>120 PRINT 15.7;STR$(15.7)
>130 PRINT STR$(VAL("34.8"))
>140 END
>RUN
 -26.3 -26.3
  15.7 15.7
 34.8

 ** DONE **
```

VAL - Value (Wert)

VAL *(String-Ausdruck)*

Die Value-Funktion ist die Umkehrung der oben erläuterten Stringzahl-Funktion (STR$). Ist der durch den String-Ausdruck angegebene String eine gültige Darstellung einer numerischen Konstanten, dann formt die Value-Funktion den String in eine numerische Konstante um.
Zum Beispiel: wenn A = "1234", dann ist VAL(A$) = 1234. Für die Auswertung von String-Ausdrücken werden die üblichen Regeln angewendet. Ist der angegebene String keine gültige Darstellung einer Zahl oder ein String mit der Länge Null, wird die Nachricht "BAD ARGUMENT" (falsches Argument) angezeigt und der Programmlauf unterbrochen. Wenn Sie einen String mit mehr als 254 Zeichen angeben, erscheint in der Anzeige ebenfalls die Nachricht "BAD ARGUMENT" und das Programm wird unterbrochen.

```
>NEW

>100 P$="23.6"
>110 N$="-4.7"
>120 PRINT VAL(P$);VAL(N$)
>130 PRINT VAL("52"&".5")
>140 PRINT VAL(N$&"E"&"12")
>150 PRINT STR$(VAL(P$))
>160 END
>RUN
  23.6 -4.7
  52.5
 -4.7E+12
 23.6

 ** DONE **
```

Anwenderdefinierte Funktionen

Einführung

Neben den festprogrammierten Funktionen, die in den beiden vorangehenden Abschnitten beschrieben sind, sieht das TI-BASIC auch anwenderdefinierbare Funktionen vor. Diese können die Programmierung vereinfachen, indem die wiederholte Anwendung komplizierter Ausdrücke vermieden wird.

DEFine (Definition)

DEF { *Bezeichnung f, numerische Funktion* [*(Parameter)*] = *numerischer Ausdruck*
Bezeichnung f. String-Funktion [*(Parameter)*] = *String-Ausdruck* }

Das DEF-Statement ermöglicht Ihnen die Definition Ihrer eigenen Funktionen zur Anwendung innerhalb eines Programms. Die Bezeichnung, die Sie für Ihre Funktion angeben, kann jeder beliebige, gültige Variablenname sein.

Wenn Sie nach der Funktionsbezeichnung noch einen Parameter spezifizieren, muß der Parameter in Klammern gesetzt werden und auch er kann jeden beliebigen gültigen Variablennamen annehmen. Wenn Ihr Ausdruck zu einem String-Resultat führt, muß Ihre Funktionsbezeichnung der Name für eine String-Variable sein (d.h., das letzte Zeichen ist das Dollarsymbol $).

Das DEF-Statement spezifiziert die anzuwendende Funktion, basierend auf dem Parameter (wenn angegeben), den Variablen, Konstanten und anderen eingebauten Funktionen. Sobald eine Funktion definiert ist, können Sie diese durch Eingabe der Funktionsbezeichnung in jedem String oder numerischen Ausdruck verwenden.

Der Funktionsbezeichung muß ein in Klammern gesetztes Argument folgen, wenn im DEF-Statement ein Parameter spezifiziert wurde.

Hat eine Funktion keinen spezifizierten Parameter, wenn in einem Ausdruck die Bezugnahme auf die Funktion vorkommt, dann wird die Funktion mit den augenblicklichen Variablenwerten durchgeführt, die im DEF-Statement erscheinen.

Beispiele:

```
>NEW

>100 DEF PI=4*ATN(1)
>110 PRINT COS(60*PI/180)
>120 END
>RUN
  .5

 ** DONE **
```

```
>NEW

>100 REM EVALUATE Y=X*(X-3)
>110 DEF Y=X*(X-3)
>120 PRINT " X  Y"
>130 FOR X=-2 TO 5
>140 PRINT X;Y
>150 NEXT X
>160 END
>RUN
  X  Y
 -2  10
 -1  4
  0  0
  1 -2
  2 -2
  3  0
  4  4
  5  10

 ** DONE **
```

DEFine (Definition)

Wenn Sie einen Parameter für eine Funktion angeben, und die Bezugnahme auf die Funktion kommt in einem Ausdruck vor, dann wird das Argument berechnet und sein Wert dem Parameter zugeordnet. Anschließend wird der Ausdruck im DEF-Statement mit dem neu zugeordneten Parameter-Wert und den gegenwärtigen Werten der anderen Variablen im DEF-Statement errechnet.

Beispiele:

```
>NEW

>100 REM TAKE A NAME AND
 PRINT IT BACKWARDS
>110 DEF BACK$(X)=SEG$(NAME$,
 X,1)
>120 INPUT "NAME? ":NAME$
>130 FOR I=LEN(NAME$) TO 1 ST
 EP -1
>140 BNAME$=BNAME$&BACK$(I)
>150 NEXT I
>160 PRINT NAME$:BNAME$
>170 END
>RUN
 NAME? ROBOT
 ROBOT
 TOBOR

 ** DONE **
```

DEF

Der im DEF-Statement verwendete Parameter ist auf dieses DEF-Statement begrenzt.

Das heißt, er ist klar abgegrenzt von einer Variablen mit dem gleichen Namen, die in anderen Statements im Programm verwendet wird. Auf diese Weise hat die Berechnung der Funktion keinen Einfluß auf den Wert einer Variablen, die die gleiche Bezeichnung wie der Parameter hat.

```
>NEW

>100 DEF FUNC(A)=A*(A+B-5)
>110 A=6.9
>120 B=13
>130 PRINT "B= ";B:"FUNC(3)=
 ";FUNC(3):"A= ";A
>140 END
>RUN
 B=  13
 FUNC(3)=  33
 A=  6.9

 ** DONE **
```

Ein DEF-Statement wird nur durchgeführt, wenn in einem Ausdruck Bezug auf die Funktion genommen wird, die es definiert. Wenn der Computer beim Ablauf eines Programms auf ein DEF-Statement trifft, geht er zum nächsten Statement weiter, ohne Anweisungen auszuführen. Ein DEF-Statement kann an beliebiger Stelle im Programm erscheinen, und muß der Bezugnahme auf die Funktion nicht unmittelbar aus logischen Gründen vorausgehen; aber die Definition der Funktion muß eine niedrigere Zeilennummer haben als das Statement, das sich auf diese Funktion bezieht. Ein DEF-Statement kann sich auf andere definierte Funktionen beziehen.

```
>NEW

>100 REM FIND F'(X) USING
 NUMERICAL APPROXIMATION
>110 INPUT "X=? ":X
>120 IF ABS(X)>.01 THEN 150
>130 H=.00001
>140 GOTO 180
>150 H=.001*ABS(X)
>160 DEF F(Z)=3*Z^3-2*Z+1
>170 DEF DER(X)=(F(X+H)-F(X-H
 ))/(2*H)
>180 PRINT "F'(";STR$(X);")=
 ";DER(X)
>190 END
>RUN
 X=? .1
 F'(.1)= -1.90999997

 ** DONE **
```

DEF

In einem DEF-Statement kann die Funktion, die Sie spezifizieren, weder direkt (zum Beispiel: DEF B = B*2) noch indirekt (z.B.: DEF F = G; DEF G = F) auf sich selbst verweisen. Der von Ihnen angegebene Parameter kann nicht als Datenfeld verwendet werden. Man kann jedoch in einer Funktionsdefinition ein Feldelement einsetzen, solange das Datenfeld nicht die gleiche Bezeichnung trägt wie der Parameter.

Wenn Sie bei der Definition einer Funktion einen Parameter spezifizieren, müssen Sie bei der Bezugnahme auf die Funktion ein Argument angeben. Umgekehrt gilt: wenn Sie bei der Definition einer Funktion keinen Parameter spezifizieren, können Sie bei der Bezugnahme auf die Funktion kein Argument angeben.

Beispiele:

```
>NEW

>100 DEF GX(X)=GX(2)*X
>110 PRINT GX(3)
>120 END
>RUN

 * MEMORY FULL IN 110

>100 DEF GX(A)=A(3)^2
>RUN

 * NAME CONFLICT IN 100
```

```
>NEW

>100 DEF SQUARE(X)=X*X
>110 PRINT SQUARE
>120 END
>RUN

 * NAME CONFLICT IN 110

>100 DEF PI=3.1416
>110 PRINT PI(2)
>RUN

 * NAME CONFLICT IN 110
```

Datenfelder (engl.: Arrays)

Einführung

Ein Datenfeld ist eine Sammlung von Variablen, die so angeordnet sind, daß man sie problemlos in einem Computerprogramm verwenden kann. Die gebräuchlichste Art ist, die Variablen in einer Liste zu gruppieren, die man als eindimensionales Datenfeld bezeichnet. Jede Variable in der Liste ist ein Feldelement (Element des Datenfeldes). Die Länge der Liste ist nur durch den verfügbaren Speicherplatz begrenzt.

Durch Anwendung der Datenfeldmöglichkeiten des TI-BASIC haben Sie mit einer Liste erheblichen Spielraum – Sie können die Elemente vorwärts oder rückwärts ausdrucken lassen, sie neu ordnen, addieren, multiplizieren oder bestimmte Elemente für die Verarbeitung auswählen.
Im TI-BASIC kann ein Datenfeld mit Element 0 oder Element 1 beginnen. Durch Anwendung des OPTION BASE Statements steuern Sie, welches Anfangselement der Computer aufstellt.

Um die Beschreibung der Datenfelder nicht zu unterbrechen, gehen wir hier davon aus, daß das erste Element in jedem Datenfeld das Element 1 ist.

Z. B. können Sie den Computer nutzen, um jede mögliche Zahlenkombination aus 2 Listen mit je 4 Zahlen ausdrucken zu lassen. Sie bezeichnen das erste Datenfeld mit X und das zweite mit Y. Da X und Y die Namen für eine Zahlen*sammlung* sind, und nicht für eine einzelne Variable, braucht der Computer eine Möglichkeit, auf die Einzelelemente in jedem Datenfeld Bezug zu nehmen. Sie müssen dem bestimmten Element im Datenfeld, das der Computer verwenden soll, einen Zeiger, einen sogenannten Index beigeben. Dieser Index wird in Klammern gesetzt und folgt unmittelbar nach der Bezeichnung des Datenfeldes. Der Index kann explizit sein, wie etwa X(3), der auf das dritte Element in der Liste X verweist, oder er ist eine Variable, wie in X(T), wo der Wert von T das richtige Element aufzeigt. In jedem Fall ist der Index immer eine positive ganze Zahl oder null.

Das Programm rechts paart die Zahlen im Datenfeld X und im Feld Y. Beachten Sie, daß mit der Datenfeld-Technik für dieses relativ komplexe Verfahren nur einige Programmzeilen erforderlich sind.

Beispiele:

```
>NEW

>100 REM THIS PROGRAM PAIRS
 TWO LISTS
>110 REM LINES 120 TO 150
 ASSIGN VALUES TO LIST X
>120 FOR T=1 TO 4
>130 READ X(T)
>140 NEXT T
>150 DATA 1,3,5,7
>160 REM LINES 170 TO 200
 ASSIGN VALUES TO LIST Y
>170 FOR S=1 TO 4
>180 READ Y(S)
>190 NEXT S
>200 DATA 2,4,6,8
>210 REM LINES 220 TO 270
 PAIR THE LISTS AND PRINT
 THE COMBINATIONS
>220 FOR T=1 TO 4
>230 FOR S=1 TO 4
>240 PRINT X(T);Y(S);" ";
>250 NEXT S
>260 PRINT
>270 NEXT T
>280 END
>RUN
  1  2   1  4   1  6   1  8
  3  2   3  4   3  6   3  8
  5  2   5  4   5  6   5  8
  7  2   7  4   7  6   7  8

 ** DONE **
```

DIMension

DIM { *Datenfeldname (ganze Zahl 1* [, *ganze Zahl 2*][, *ganze Zahl 3*] } ..

Das DIM-Statement (Dimensions-Statement) reserviert Raum für numerische und String-Datenfelder. In Ihrem Programm können Sie ein Datenfeld nur einmal in expliziter Form dimensionieren. Wenn Sie es tun, muß das DIM-Statement im Programm vor jeder anderen Bezugnahme auf das Datenfeld erscheinen.

Bei der Dimensionierung von mehr als einem Datenfeld in einem einzelnen DIM-Statement sind die Datenfeldnamen durch Kommas zu trennen. Der Datenfeldname kann ein beliebiger gültiger Variablenname sein.

Sie haben die Wahl, in TI-BASIC mit ein-, zwei- oder dreidimensionalen Datenfeldern und ihren Anwendungen zu operieren. Die Anzahl der Werte in Klammern nach dem Datenfeldnamen informiert den Computer, wie viele Dimensionen das Datenfeld aufweist.

Eindimensionale Felder haben nur einen ganzzahligen Wert nach dem Namen. Zweidimensionale Felder werden mit zwei ganzzahligen Werten beschrieben, die die Zeilen-und Spaltenzahl bestimmen.

Dreidimensionale Felder benötigen drei ganzzahlige Werte für die Definition ihrer Eigenschaften.

- DIM A(6) - beschreibt ein eindimensionales Datenfeld;
- DIM A(12,3) - beschreibt ein zweidimensionales Datenfeld;
- DIM A(5,2,11) - beschreibt ein dreidimensionales Datenfeld;

Wird das Datenfeld in einem DIM-Statement nicht dimensioniert, ordnet der Computer automatisch einen Wert von 10 für die ganze Zahl 1 zu (und bei Bedarf ebenfalls 10 für die ganze Zahl 2 und 3). Dies gilt für jedes verwendete Datenfeld.

Der Raum für Ihr Datenfeld wird nach Eingabe des RUN-Befehls zugewiesen, ehe das Programm dann tatsächlich abläuft. Jedes Element in einem Stringfeld ist jedoch ein Nullstring, bis Sie tatsächlich bei jedem Element Werte einsetzen. Wenn Ihr Computerspeicher ein Datenfeld mit den angegebenen Dimensionen nicht verarbeiten kann, erhalten Sie die Nachricht "MEMORY FULL" (Speicher belegt) und das Programm wird nicht durchgeführt.

Beispiele:

```
>DIM A(12),B(5)
```

```
>NEW

>100 DIM X(15)
>110 FOR I=1 TO 15
>120 READ X(I)
>130 NEXT I
>140 REM PRINT LOOP
>150 FOR I=15 TO 1 STEP -1
>160 PRINT X(I);
>170 NEXT I
>180 DATA 1,2,3,4,5,6,7,8,9,1
 0,11,12,13,14,15
>190 END
>RUN
  15  14  13  12  11  10  9
  8  7  6  5  4  3  2  1
 ** DONE **
```

DIM

Indizieren eines Datenfeldes

Jedesmal, wenn Sie auf ein Datenfeld in Ihrem Programm verweisen wollen, müssen Sie genau angeben, welches Feldelement der Computer verwenden soll. Zu diesem Zweck zeigen Sie mit einem Index auf das Element. Indices werden in Klammern gesetzt und stehen unmittelbar nach dem Feldnamen. Ein Index kann jeder beliebige gültige numerische Ausdruck sein, der zu einem nicht-negativen Resultat führt. Dieses Resultat wird bei Bedarf auf die nächste ganze Zahl gerundet.

Die Zahl der für ein Datenfeld reservierten Elemente bestimmt den maximalen Wert jedes Index für dieses Datenfeld. Bei Anwendung eines Datenfelds, das nicht in einem DIM-Statement definiert ist, ist der Maximalwert für jeden Index 10. Der Minimalwert ist 0, wenn nicht ein OPTION BASE Statement den minimalen Indexwert auf 1 setzt. Auf diese Weise hat ein als DIM A(6) definiertes Datenfeld tatsächlich sieben zugängliche Elemente in TI-BASIC, wenn nicht der Null-Index durch das OPTION BASE 1 Statement eliminiert wurde.

Das Beispiel rechts geht davon aus, daß das Datenfeld mit Element 1 beginnt (OPTION BASE 1 auf Zeile 120):

- Zeile 130 - Diese Zeile definiert T als eindimensionales Feld mit 25 Feldelementen.
- Zeile 160 - Die numerische Variable I indiziert hier T. Der jeweilige Wert, den I zu dieser Zeit enthält, wird verwendet, um auf ein Element von T zu zeigen. Wenn I = 3, wird das dritte Element von T addiert.
- Zeile 200 - Der Index 14 weist den Computer an, das vierzehnte Element von T auszudrucken.
- Zeile 220 - Der Computer berechnet den numerischen Ausdruck N + 2. Ist N zu dieser Zeit 15, wird das 17. Element von T ausgedruckt.

Wenn Sie den Zugriff auf ein Datenfeld mit einem Index versuchen, der das Maximum der Elemente überschreitet, die für dieses Feld definiert sind, oder wenn Ihr Index bei Anwendung des OPTION BASE 1 Statements einen Wert von null hat, wird die Nachricht "BAD SUBSCRIPT" (falscher Index) ausgedruckt, und das Programm unterbrochen.

Beispiele:

```
>NEW

>100 REM DEMO OF DIM AND
 SUBSCRIPTS
>110 S=100
>120 OPTION BASE 1
>130 DIM T(25)
>140 FOR I=1 TO 25
>150 READ T(I)
>160 A=S+T(I)
>170 PRINT A;
>180 NEXT I
>190 PRINT::
>200 PRINT T(14)
>210 INPUT "ENTER A NUMBER BE
 TWEEN 1 AND 23:":N
>220 PRINT T(N+2)
>230 DATA 12,13,43,45,65,76,7
 8,98,56,34,23,21,100,333,222
 ,111,444,666,543,234,89,765,
 90,101,345
>240 END
>RUN
  112  113  143  145  165
  176  178  198  156  134
  123  121  200  443  322
  211  544  766  643  334
  189  865  190  201  445

  333
 ENTER A NUMBER BETWEEN 1 AND
  23:14
  111

 ** DONE **
```

OPTION BASE

OPTION BASE $\left\{\begin{matrix}0\\1\end{matrix}\right\}$

Das OPTION BASE Statement gibt Ihnen die Möglichkeit, die untere Grenze der Feldindices auf 1 anstatt auf 0 zu setzen. Das OPTION BASE Statement kann ausgelassen werden, wenn die Untergrenze der Indices null sein soll.

Wenn Sie das OPTION BASE Statement in Ihr Programm aufnehmen, muß es eine niedrigere Zeilennummer als jedes DIM-Statement oder jede Bezugnahme auf ein Feldelement erhalten. In einem Programm ist nur ein OPTION BASE Statement zulässig, und es gilt für die Indices aller Datenfelder im Programm.

Es ist also nicht möglich, daß in einem Programm ein Feldindex mit null beginnt und ein anderer mit eins.

Wenn Sie im OPTION BASE Statement eine andere ganze Zahl als null oder eins verwenden, stoppt der Computer das Programm und druckt die Nachricht "INCORRECT STATEMENT" (Statement nicht korrekt) aus.

Beispiele:

```
>NEW

>100 OPTION BASE 1
>110 DIM X(5,5,5)
>120 X(1,0,1)=3
>130 PRINT X(1,0,1)
>140 END
>RUN

 * BAD SUBSCRIPT IN 120

>100 ENTER
>RUN
  3

 ** DONE **
```

Unterprogrammme (Subroutinen)

Einführung

Unterprogramme kann man sich als separate, in sich geschlossene Programme innerhalb eines Hauptprogramms vorstellen. Sie führen gewöhnlich eine bestimmte Aktion aus, wie den Ausdruck von Informationen, Berechnungen, oder das Einlesen von Werten in ein Datenfeld. Setzt man diese Aktionen in ein Unterprogramm, braucht man diese Gruppe von Statements nur einmal einzutippen, und kann sie dann von beliebiger Stelle aus in einem Programm mit dem GOSUB-Statement durchführen.

Zu Beginn verhält sich das GOSUB-Statement wie GOTO.

Es veranlaßt, daß der Computer zu der aufgelisteten Zeilennummer springt. Die Subroutinenprogrammierung verleiht dem Computer jedoch die Fähigkeit, "sich zu erinnern", wo die Verzweigung im Hauptprogramm stattfand, und nach Beendigung des Unterprogramms an diese Stelle zurückzukehren. Dieses Verfahren erfordert, daß das Unterprogramm mit einem RETURN-Statement abgeschlossen wird. Im allgemeinen steht im Programm entweder ein STOP-Statement oder eine andere unbedingte Verzweigungsanweisung unmittelbar vor den Unterprogrammen, so daß der Computer nicht zufällig in die Unterprogramme "eindringt". Die Unterprogramme dürfen nur mit dem GOSUB-Befehl eingeleitet und können an beliebiger Zeilennummer innerhalb der Subroutine begonnen werden.

GOSUB

GOSUB / **GO SUB** *Zeilennummer*

Das GOSUB-Statement verwendet man zusammen mit dem RETURN-Statement, um das Verzweigen des Programms zu einem Unterprogramm zu ermöglichen, die Schritte im Unterprogramm abzuschließen, und zur Programmzeile nach dem GOSUB-Statement zurückzukehren. Wenn der Computer ein GOSUB-Statement durchführt, speichert er die nächste Zeilennummer des Hauptprogramms, um beim Auffinden eines RETURN-Statements im Unterprogramm an diese Stelle zurückkehren zu können.
(Es ist Ihrer Wahl überlassen, ob Sie zwischen GO und SUB eine Leerstelle wünschen).

Beispiele:

```
>NEW

>100 REM BUILD AN ARRAY,
 MULTIPLY EACH ELEMENT BY 3,
 PRINT BOTH ARRAYS
>110 FOR X=1 TO 4
>120 FOR Y=1 TO 7
>130 I(X,Y)=INT(30*RND)+1
>140 NEXT Y
>150 NEXT X
>160 PRINT "FIRST ARRAY":
>170 GOSUB 260
>180 FOR X=1 TO 4
>190 FOR Y=1 TO 7
>200 I(X,Y)=3*I(X,Y)
>210 NEXT Y
>220 NEXT X
>230 PRINT "3 TIMES VALUES IN
 FIRST ARRAY":
>240 GOSUB 260
>250 STOP
>260 REM SUBROUTINE TO PRINT
 ARRAY
>270 FOR X=1 TO 4
>280 FOR Y=1 TO 7
>290 PRINT I(X,Y);
>300 NEXT Y
>310 PRINT
>320 NEXT X
>330 PRINT
>340 RETURN
>RUN
 FIRST ARRAY

  16  12  17  12  8  17  8
  18  22  1  29  16  14  11
  5  25  22  4  24  11  24
  26  21  18  2  12  20  15

 3 TIMES VALUES IN FIRST ARRA
 Y

  48  36  51  36  24  51  24
  54  66  3  87  48  42  33
  15  75  66  12  72  33  72
  78  63  54  6  36  60  45

 ** DONE **
```

GOSUB

Innerhalb eines Unterprogramms wünschen Sie vielleicht, daß der Computer zu einem anderen Unterprogramm springt, es beendet, dann wieder zurück zum ersten Unterprogramm geht, dort die restlichen Schritte durchläuft, und schließlich zum Hauptprogramm an die Stelle zurückkehrt, wo die ursprüngliche Verzweigung stattfand. Diesen Vorgang können Sie problemlos mit der entsprechenden Paarung von GOSUB-und RETURN-Statements realisieren. Gehen Sie jedoch bei der Entwicklung von Unterprogrammen sorgfältig vor, damit die Orientierung für den Computer erhalten bleibt.

Im Beispiel rechts springt das Hauptprogramm zum Unterprogramm 1, wenn es Zeile 500 erreicht. Bei Erreichen von Zeile 730 im Unterprogramm 1 erfolgt eine Verzweigung zum Unterprogramm 2. Wenn RETURN im Unterprogramm 2 aufgefunden wird (Zeile 850), kommt der Computer bei Zeile 740 zu Unterprogramm 1 zurück, beendet dieses Unterprogramm, geht zum Hauptprogramm und schließt es bis Zeile 600 ab.

Wird mit dem GOSUB-Statement eine Verzweigung zu einer Zeilennummer angegeben, die im vorliegenden Programm nicht existiert, stoppt das Programm und der Computer druckt die Nachricht "BAD LINE NUMBER" (falsche Zeilennummer). Wenn das GOSUB-Statement das Programm zu seiner eigenen Zeilennummer verzweigt, wird das Programm ebenfalls unterbrochen und die Nachricht "MEMORY FULL" (Speicher belegt) ausgedruckt.

Beispiele:

```
>NEW

>100 REM NESTED SUBROUTINES
>110 REM MAIN PROGRAM
      .
      .
      .
>500 GOSUB 700
>510 .
      .
      .
>600 STOP
>700 REM SUBROUTINE1
      .
      .
      .
>730 GOSUB 800
>740 .
      .
      .
>790 RETURN
>800 REM SUBROUTINE2
      .
      .
      .
>850 RETURN
```

```
>NEW

>100 X=12
>110 Y=23
>120 GOSUB 120
>130 PRINT Z
>140 STOP
>150 REM SUBROUTINE
>160 Z=X+Y*120/5
>170 RETURN
>RUN

* MEMORY FULL IN 120


>120 GOSUB 150
>RUN
  564

 ** DONE **
```

RETURN

RETURN

Das RETURN-Statement verwendet man in Verbindung mit dem GOSUB-Statement, um für das TI-BASIC eine Verzweigungs- und Rückkehr-Struktur zu bilden. Wenn der Computer auf ein RETURN-Statement trifft, gibt er die Programmsteuerung zu der Zeile unmittelbar nach dem GOSUB-Statement zurück, das den Ablauf zu dem speziellen Unterprogramm an der ersten Stelle verzweigte. Sie können ohne Schwierigkeiten Programme mit Unterprogrammen entwickeln, die zu anderen Unterprogrammen und dann wieder zurückspringen, wenn Sie darauf achten, daß jedes GOSUB-Statement den Computer zu einem RETURN-Statement führt

Wenn der Computer beim Ablauf eines Programms noch vor der Durchführung einer GOSUB-Instruktion auf ein RETURN-Statement trifft, wird das Programm mit der Nachricht "CAN'T DO THAT" (Durchführung nicht möglich) beendet.

Beispiele:

```
>NEW

>100 FOR I=1 TO 3
>110 GOSUB 150
>120 PRINT "I=";I
>130 NEXT I
>140 STOP
>150 REM SUBROUTINE
>160 FOR X=1 TO 2
>170 PRINT "X=";X
>180 NEXT X
>190 RETURN
>RUN
 X= 1
 X= 2
 I= 1
 X= 1
 X= 2
 I= 2
 X= 1
 X= 2
 I= 3

 ** DONE **
```

ON-GOSUB

ON *numerischer Ausdruck* { **GOSUB** / **GO SUB** } *Zeilennummer*[,*Zeilennummer*]...

Das ON-GOSUB-Statement verwendet man in Verbindung mit dem RETURN-Statement um den Computer anzuweisen, eines von mehreren Unterprogrammen, abhängig vom Wert eines numerischen Ausdrucks, durchzuführen, und dann zur Hauptprogrammfolge zurückzugehen.

Der Computer berechnet zuerst den numerischen Ausdruck und wandelt bei Bedarf das Resultat durch Runden in eine ganze Zahl um. Diese ganze Zahl weist das Programm an, welche Unterprogramm-Zeilennummer im ON-GOSUB-Statement als nächste durchzuführen ist. Ist der Wert des numerischen Ausdrucks 1, geht der Computer zu der Zeilennummer, die im ON-GOSUB-Statement an erster Stelle aufgelistet ist. Ist der Wert zwei, verzweigt der Computer zur zweiten angegebenen Zeilennummer etc.

Zusätzlich speichert der Computer die nächste Zeilennummer nach dem ON-GOSUB-Statement, und kehrt nach der Durchführung des Unterprogramms an diese Stelle zurück. Das Unterprogramm muß ein RETURN-Statement enthalten, das dem Computer die Rückkehr zu der gespeicherten Zeilennummer und die Fortsetzung des Programms von diesem Statement an signalisiert.

Andernfalls wird der Ablauf des Programms bis zum Ende fortgeführt, als ob anstelle des GOSUB ein GOTO-Statement durchgeführt worden wäre.

Beispiele:

```
>NEW

>100 INPUT "CODE=?":CODE
>110 IF CODE=9 THEN 290
>120 INPUT "HOURS=?":HOURS
>130 ON CODE GOSUB 170,200,23
 0,260
>140 PAY=RATE*HOURS+BASEPAY
>150 PRINT "PAY IS $";PAY
>160 GOTO 100
>170 RATE=3.10
>180 BASEPAY=5
>190 RETURN
>200 RATE=4.25
>210 BASEPAY=25
>220 RETURN
>230 RATE=10
>240 BASEPAY=50
>250 RETURN
>260 RATE=25
>270 BASEPAY=100
>280 RETURN
>290 END
>RUN
 CODE=?4
 HOURS=?40
 PAY IS $ 1100
 CODE=?2
 HOURS=?37
 PAY IS $ 182.25
 CODE=?3
 HOURS=?35.75
 PAY IS $ 407.5
 CODE=?1
 HOURS=?40
 PAY IS $ 129
 CODE=?9

 ** DONE **
```

Ist der gerundete Wert des numerischen Ausdrucks kleiner als 1 oder größer als die Zahl der Zeilennummern im ON-GOSUB-Statement, endet das Programm mit der Nachricht "BAD VALUE IN xx" (falscher Wert in xx).

```
>RUN
 CODE=?5
 HOURS=?40

 * BAD VALUE IN 130
```

Ist die aufgelistete Zeilennummer keine gültige Programmzeile, wird bei der Durchführung des Statements die Nachricht "BAD LINE NUMBER" (falsche Zeilennummer) ausgedruckt.

```
>130 ON CODE GOSUB 170,200,23
 0,600
>RUN
 CODE=?4
 HOURS=?40

 * BAD LINE NUMBER IN 130
```

Dateiverarbeitung

Einführung

Ihr Home-Computer bietet die Möglichkeit, Programme und Daten auf Peripheriegeräten zu speichern. Später können Sie diese Dateien wieder einlesen, sie beliebig oft zusammen mit Ihrem Computer anwenden, und sie löschen, wenn sie nicht mehr benötigt werden.

Dateiverarbeitung bietet Ihnen ein wirksames und vorteilhaftes Programmiermittel. Es erübrigt sich, bestehende Programme immer wieder neu einzutippen, Sie können wichtige Informationen speichern und Verfahren entwickeln, mit denen Sie Ihre wesentlichen Daten auf den neuesten Stand bringen. TI-BASIC sieht Datei-Organisation und -Verarbeitung im Direktzugriff oder seriell, feste und variable Länge der Sätze, internes Format sowie Anzeigeformat für die Daten vor. Dieser Abschnitt behandelt die Statements zur Datenverwaltung in TI-BASIC: OPEN, CLOSE, INPUT, PRINT und RESTORE. Wenn neue Peripheriegeräte auf den Markt kommen, werden deren Datei-Verwaltungsfunktionen in den entsprechenden Bedienungsanleitungen beschrieben.

Anmerkung: Bezeichnungen für Peripheriegeräte erfordern bei TI-BASIC im allgemeinen Großbuchstaben, zum Beispiel:

CS1
RS232

OPEN

OPEN #*Dateinummer: Dateibezeichnung* [,*Dateiorganisation*] [,*Dateityp*] [,*Eröffnungs-Modus*] [,*Aufzeichnungsform*] [,*Lebensdauer*]

Das OPEN-Statement stellt die erforderliche Verbindung zwischen einer in Ihrem Programm verwendeten Dateinummer und der Datei auf einem speziellen Peripheriegerät her.

Das OPEN-Statement beschreibt dem Computer die Eigenschaften einer Datei, so daß Ihr Programm die Datei verarbeiten oder bilden kann. Bei bestimmten Peripheriegeräten kontrolliert der Computer, ob die Datei oder Geräteeigenschaften mit den Informationen übereinstimmen, die im OPEN-Statement für diese Datei spezifiziert sind. Wenn sie nicht zusammenpassen, oder wenn der Computer die Datei nicht finden oder bilden kann, wird die Datei nicht eröffnet und eine E/A-Fehlernachricht ausgedruckt (siehe hierzu die Tabelle mit den Fehlernachrichten).

Dateinummer und Dateibezeichnung müssen in das OPEN-Statement aufgenommen werden. Die anderen Informationen kann man in beliebiger Reihenfolge beifügen oder ganz weglassen. Wenn Sie eine Spezifikation nicht angeben, setzt der Computer bestimmte Eigenschaften für die Datei voraus, sogenannte Standardattribute, wie später in diesem Abschnitt beschrieben.

- **Dateinummer** - Alle TI-BASIC-Statements, die auf Dateien verweisen, akzeptieren eine Dateinummer zwischen 0 und 255 inklusive. Die Dateinummer wird einer bestimmten Datei durch das OPEN-Statement zugeordnet. Da sich die Dateinummer 0 auf die Tastatur und den Bildschirm Ihres Computers bezieht und immer zugänglich ist, können Sie in Ihren Programm-Statements keine Datei mit der Dateinummer 0 eröffnen oder abschließen. Die anderen Nummern dürfen nach Belieben zugeordnet werden, solange jede offene Datei in Ihrem Programm eine unterschiedliche Nummer erhält.

  Die Dateinummer wird als Nummernzeichen (#), gefolgt von einem numerischen Ausdruck, eingegeben. Wenn der Computer diesen Ausdruck berechnet, und das Resultat auf die nächste ganze Zahl rundet, muß die Zahl zwischen 1 und 255 inklusive liegen und darf nicht mit einer anderen Dateinummer, die Sie augenblicklich in Ihrem Programm verwenden, identisch sein.
- **Dateibezeichnung** - Eine Dateibezeichnung bezieht sich, abhängig von der Art des Peripheriegerätes, entweder auf das Peripheriegerät selbst oder auf eine Datei, die auf diesem Gerät gespeichert ist. Jedes Peripheriegerät hat eine feste Bezeichnung, die der Computer erkennt. Die gültigen Dateibezeichnungen für die zwei Kassettenrekorder sind zum Beispiel "CS1" und "CS2". Durch Aufnahme der Dateibezeichnung in das OPEN-Statement weisen Sie den Computer an, den Zugriff auf eine bestimmte Datei oder ein Peripheriegerät durchzuführen, wann immer im Programm auf die entsprechende Dateibezeichnung verwiesen ist. Als Dateibezeichnung kann jeder beliebige String-Ausdruck verwendet werden, der in der Auswertung eine zulässige Dateibezeichnung ergibt. Bei Anwendung einer String-Konstanten muß diese in Anführungszeichen gesetzt werden.

Beispiele:

```
>100 OPEN #2:"CS1",SEQUENTIAL
 ,INTERNAL,INPUT,FIXED 128,PE
 RMANENT
```

```
>100 OPEN #25:"CS1",SEQUENTIA
 L,INTERNAL,INPUT,FIXED,PERMA
 NENT
>110 X=100
>120 OPEN #X+5:"CS2",SEQUENTI
 AL,INTERNAL,OUTPUT,FIXED,PER
 MANENT
```

```
>130 N=2
>140 OPEN #122:"CS"&STR$(N),S
 EQUENTIAL,INTERNAL,OUTPUT,FI
 XED,PERMANENT
```

OPEN

Informationen über Dateibezeichnungen (Namen) in Verbindung mit dem TI Disketten-System, der V.24 Schnittstelle und anderen Peripheriegeräten erhalten Sie in den jeweiligen Bedienungsanleitungen.

- **Dateiorganisation** - Die in TI-BASIC verwendeten Dateien können entweder sequentiell oder für wahlfreien (Direkt-) Zugriff organisiert sein. Aufzeichnungen einer sequentiell organisierten Datei werden nacheinander in der Folge von Anfang bis Ende gelesen oder geschrieben. Die Dateien mit direktem Zugriff können in jeder beliebigen Aufzeichnungsfolge gelesen oder geschrieben werden (in TI-BASIC finden Sie dafür die Bezeichnung RELATIVE). Eine Verarbeitung ist ebenfalls möglich.

  Zur Angabe der logischen Struktur einer Datei geben Sie im OPEN-Statement entweder SEQUENTIAL oder RELATIVE ein. (SEQUENTIAL entspricht sequentiell, RELATIVE entspricht dem wahlfreien Zugriff). Es steht Ihnen frei, die Anfangszahl der Sätze in einer Datei dadurch zu spezifizieren, daß Sie dem Wort SEQUENTIAL oder RELATIVE noch einen numerischen Ausdruck nachstellen.

  Wenn Sie die Angaben zur Dateiorganisation auslassen, setzt den Computer eine sequentielle Organisation (SEQUENTIAL) voraus.

- **Dateityp** - Diese Spezifikation gibt das Format der Daten an, die auf der Datei gespeichert sind: DISPLAY (Anzeige) oder INTERNAL (intern).

  DISPLAY-Format bezieht sich auf druckbare Zeichen (ASCII). Im allgemeinen verwendet man das DISPLAY-Format, wenn die Ausgabeinformationen von Personen und nicht vom Computer gelesen werden. Jede DISPLAY-Aufzeichnung entspricht gewöhnlich einer Druckzeile.

INTERNAL-Daten werden im internen Maschinenformat aufgezeichnet, das nicht in druckbare Zeichen übersetzt wurde. Daten in dieser Form können problemlos vom Computer, nicht aber von Personen gelesen werden. (Eine Erklärung der internen Speicherung der Daten finden Sie im Abschnitt „INPUT".)

INTERNAL-Format ist für die Aufzeichnung der Daten auf einem Speichergerät wie einer Kassette effektiver. Es benötigt weniger Speicherplatz und ist leichter mit einem PRINT-Statement zu formatieren. (Im Abschnitt „PRINT" finden Sie Anweisungen über die Formatierung von PRINT-Statements für INTERNAL-Aufzeichnungen und DISPLAY-Aufzeichnungen.)

Da der Computer INTERNAL-Daten intern verwendet, läuft das Programm schneller, wenn Ihre Dateien im INTERNAL-Format gehalten sind. Der Computer muß in diesem Fall nicht DISPLAY-Zeichen in INTERNAL-Format übertragen und wieder für die Anzeige umformen.

Wenn Sie diese Spezifikation weglassen, setzt der Computer DISPLAY-Format voraus.

Beispiele:

```
>100 OPEN #4:"CS2",OUTPUT,INT
 ERNAL,SEQUENTIAL,FIXED
```

```
>120 OPEN #12:NAME$,RELATIVE
 50,INPUT,FIXED,INTERNAL
```

```
>100 OPEN #10:"CS1",OUTPUT,FI
 XED
```

(Computer nimmt SEQUENTIAL, DISPLAY, PERMANENT an)

OPEN

- **Open-Modus** - Diese Eingabe weist den Computer an, die Datei im INPUT-, OUTPUT-, UPDATE- oder APPEND-Modus zu verarbeiten. Wenn Sie diese Angabe weglassen, setzt der Computer den UPDATE-Modus voraus.
  - INPUT-Dateien (Eingabedateien) können nur gelesen werden.
  - OUTPUT-Dateien (Ausgabedateien) können nur geschrieben werden. Die neu gebildete Datei hat dann alle Eigenschaften, die in den Spezifikationen des OPEN-Statements angeführt sind, bzw. alle Standardattribute.
  - UPDATE-Dateien (Änderungs- oder Bewegungsdateien) können gelesen und geschrieben werden. Die übliche Verarbeitung ist, eine Aufzeichnung zu lesen, sie in irgendeiner Weise zu ändern, und den geänderten Satz wieder auf die Datei zu überschreiben.
  - Der APPEND-Modus erlaubt das Anfügen von Daten am Ende der existierenden Datei. In diesem Modus ist der Zugriff auf die bereits vorhandenen Sätze der Datei nicht möglich.
- **Satztyp** - Diese Eingabe spezifiziert ob die Sätze auf der Datei alle die gleiche Länge haben (FIXED), oder ob sie in der Länge variieren (VARIABLE). Dem Schlüsselwort FIXED oder VARIABLE kann ein numerischer Ausdruck folgen, der die Maximallänge eines Satzes angibt. Jedes Peripheriegerät hat eine maximale Satzlänge; achten Sie also auf die entsprechenden Bedienungsanleitungen. Wenn Sie die Längenspezifikationen auslassen, nimmt der Computer eine vom jeweiligen Peripheriegerät abhängige Satzlänge an.

  Wird eine Datei mit RELATIVE spezifiziert, müssen Sie bei den Sätzen eine feste Länge wählen (FIXED). Wird diese Eingabe für die Direktzugriffs-Dateien ausgelassen, setzt der Computer Sätze mit fester Länge voraus, wobei die tatsächliche Länge vom jeweiligen Peripheriegerät abhängt.

  Sequentiell organisierte Dateien können feste (FIXED) oder variable (VARIABLE) Satzlängen haben. Wenn man diese Spezifikation für die sequentiellen Dateien ausläßt, geht der Computer von Sätzen mit variabler Länge aus.

  Werden die Sätze mit FIXED spezifiziert, füllt der Computer jeden Satz auf der rechten Seite auf, um sicherzustellen, daß die spezifizierte Länge gegeben ist. Wenn die Daten im DISPLAY-Format aufgenommen werden, füllt der Computer den Satz mit Leerstellen. Bei Verwendung des INTERNAL-Formats wird der Satz mit fester Länge mit Binärnullen aufgefüllt.
- **Dauer** – Dateien, die Sie mit Ihrem Home-Computer erstellen, werden als permanent (PERMANENT) und nicht als temporär betrachtet. Sie können die Eingabe der Dauer weglassen, da der Computer automatisch eine permanente Dateidauer annimmt.

Beispiele:

```
>100 OPEN #53:NAME$,FIXED,INT
 ERNAL,RELATIVE

 (Computer nimmt UPDATE an)
```

```
>100 OPEN #11:NAME$,INPUT,INT
 ERNAL,SEQUENTIAL,VARIABLE 10
 0

>100 OPEN #75:"CS1",OUTPUT,FI
 XED

 (Computer nimmt SEQUENTIAL,
  DISPLAY, FIXED und eine
  Länge von 64 Stellen an)
```

OPEN

Kassettenrekorder-Informationen

- **Dateinummer*** - jede Zahl zwischen 1 und 255 inklusive
- **Dateibezeichnung*** - "CS1" oder "CS2"
- **Dateiorganisation** - SEQUENTIAL (seriell)
- **Dateityp** - INTERNAL (bevorzugt) oder DISPLAY
- **Open-Modus*** - INPUT oder OUTPUT
- **Satztyp*** - FIXED (feste Länge)

*Diese Spezifikation ist erforderlich.

Für Aufnahmen auf dem Kassettenrekorder können Sie eine beliebige Länge bis maximal 192 Stellen angeben. Das Kassettengerät verwendet dagegen Sätze mit 64, 128 oder 192 Stellen, und füllt den von Ihnen spezifizierten Satz bis zur entsprechenden Länge auf. Wenn Sie also eine 83stellige Kassettenaufzeichnung angeben, schreibt der Computer tatsächlich einen 128stelligen Satz ein. Ist die Länge nicht spezifiziert, wird eine 64stellige Satzlänge angenommen. Bei Kassettengeräten vergleicht der Computer nicht die Dateispezifikationen im OPEN-Statement mit den Eigenschaften einer existierenden Datei.

Wenn der Computer das OPEN-Statement für ein Kassettengerät durchführt, erhalten Sie für die Aktivierung des Kassettenrekorders die auf der rechten Seite erscheinenden Instruktionen auf dem Bildschirm. Bitte beachten: Nur "CS1" dient zum Laden in den Computer. Zum Speichern kann "CS1" wie "CS2" gleichermaßen benutzt werden.

Beispiele:

```
>NEW

>100 OPEN #2:"CS1",INTERNAL,I
 NPUT,FIXED
 .
 . Programmzeilen
 .
>290 CLOSE #2
>300 END
>RUN

 * REWIND CASSETTE TAPE    CS1
   THEN PRESS ENTER

 * PRESS CASSETTE PLAY     CS1
   THEN PRESS ENTER
   .
   . Rest des Programmablaufs
   .
 * PRESS CASSETTE STOP     CS1
   THEN PRESS ENTER

 ** DONE **
```

CLOSE

CLOSE # *Dateinummer* **[:DELETE]**

Das CLOSE-Statement schließt die Datei ab, oder unterbricht die Verbindung zwischen Datei und Programm. Nach Durchführung des CLOSE-Statements steht die "abgeschlossene" Datei für Ihr Programm nicht mehr zur Verfügung, wenn Sie nicht wieder eine OPEN-Anweisung geben. Der Computer wird auch die abgeschlossene Datei nicht mehr länger mit der Dateinummer in Verbindung bringen, die Sie in Ihrem Programm angegeben haben. Sie können dann diese Dateinummer wieder jeder beliebigen anderen Datei zuordnen.

Wenn Sie DELETE im CLOSE-Statement wählen, hängt die nun folgende Aktion vom jeweiligen Peripheriegerät ab. Sie wird in den entsprechenden Bedienungsanleitungen beschrieben.

Bei dem Versuch, eine Datei abzuschließen, die zuvor in dem Programm nicht eröffnet wurde, beendet der Computer das Programm mit der Nachricht "FILE ERROR" (Dateifehler).

Zum Schutz Ihrer Dateien schließt der Computer automatisch alle offenen Dateien ab, wenn ein Fehler auftritt, der zur Beendigung des Programms führt. Bei einer Stoppstelle im Programm, die durch einen BREAK-Befehl oder durch drücken von **CLEAR** verursacht wurde, werden offene Dateien automatisch nur unter folgenden Voraussetzungen abgeschlossen:

- Sie editieren das Programm
- Sie beenden den BASIC-Modus mit dem BYE-Befehl
- Sie lassen das Programm erneut ablaufen (RUN)
- Sie geben einen NEW-Befehl ein

Achtung!

Wenn Sie mit **QUIT** das Programm verlassen, schließt der Computer offene Dateien nicht ab, und Sie könnten in diesem Fall Daten verlieren. Ist das Verlassen des Datei-Verarbeitungs-Programms vor dem tatsächlichen Ende erforderlich, befolgen Sie die nachstehenden Instruktionen, um keine Daten zu verlieren.

Drücken Sie **CLEAR**, bis der Computer mit der Nachricht "BREAKPOINT AT xx" reagiert. Dieser Vorgang kann mehrere Sekunden dauern.

- Geben Sie BYE ein, wenn der Positionszeiger wieder auf dem Bildschirm erscheint.

Beispiele:

```
>NEW
>100 OPEN #6:"CS1",SEQUENTIAL
 ,INTERNAL,INPUT,FIXED
>110 OPEN #25:"CS2",SEQUENTIA
 L,INTERNAL,OUTPUT,FIXED
 .
 . Programmzeilen
 .
>200 CLOSE #6:DELETE
>210 CLOSE #25
>220 END
```

CLOSE

Kassettenrekorder-Informationen

Immer, wenn der Computer das CLOSE-Statement für einen Kassettenrekorder durchführt, erhalten Sie für die Bedienung des Rekorders die rechts aufgelisteten Instruktionen auf Ihrem Bildschirm.

Bei Anwendung der DELETE-Option mit Kassettenrekordern findet über das Abschließen der Datei hinaus keine weitere Aktion statt.

Beispiele:

```
>NEW
>100 OPEN #24:"CS1",INTERNAL,
 INPUT,FIXED
>110 OPEN #19:"CS2",INTERNAL,
 OUTPUT,FIXED
 .
 . Programmzeilen
 .
>200 CLOSE #24
>210 CLOSE #19
>220 END
>RUN

 * REWIND CASSETTE TAPE     CS1
   THEN PRESS ENTER

 * PRESS CASSETTE PLAY      CS1
   THEN PRESS ENTER

 * REWIND CASSETTE TAPE     CS2
   THEN PRESS ENTER

 * PRESS CASSETTE RECORD    CS2
   THEN PRESS ENTER
   .
   . Programm läuft
   .

 * PRESS CASSETTE STOP      CS1
   THEN PRESS ENTER

 * PRESS CASSETTE STOP      CS2
   THEN PRESS ENTER

 ** DONE **
```

INPUT

INPUT # *Dateinummer* [,*REC numerischer Ausdruck*] [:*Variablenliste*]
Siehe auch Abschnitt "INPUT-OUTPUT"-Statements (Ein-/Ausgabe-Anweisungen)

Diese Form des INPUT-Statements erlaubt Ihnen das Lesen von Daten von einem Peripheriegerät. Das INPUT-Statement kann nur zusammen mit Dateien verwendet werden, die im INPUT- oder im UPDATE-Modus eröffnet wurden. Die Dateinummer im INPUT-Statement muß die Nummer einer momentan offenen Datei sein. Verwendung der Dateinummer 0, der Tastatur, ist immer möglich. Wenn Sie diese wählen, wird das INPUT-Statement durchgeführt (wie in der Beschreibung im Abschnitt "INPUT-OUTPUT-Statements"), außer wenn Sie keinen Eingabe-Dialog spezifizieren können.

Die Variablenliste enthält Variable, denen bei der Durchführung des INPUT-Statements Werte zugeordnet werden. Variablennamen in dieser Liste werden durch Kommata getrennt und können numerische und/oder String-Variable sein.

Beispiele:

```
>NEW

>100 OPEN #13:"CS1",SEQUENTIA
 L,DISPLAY,INPUT,FIXED
>110 INPUT #13:A,B,C$,D$,X,Y,
 Z$
>120 IF A=99 THEN 150
>130 PRINT A;B:C$:D$:X;Y:Z$
>140 GOTO 110
>150 CLOSE #13
>160 END
>RUN

 --Die auf Kassette gespeicherten
   Daten werden auf dem Bildschirm
   angezeigt
 ** DONE **
```

Auffüllen der Variablenliste

Wenn der Computer die Sätze aus einer Datei einliest, speichert er jeden vollständigen Satz intern in einem temporären Speicherbereich, dem sogenannten Ein-/Ausgabepuffer (E/A-Puffer). Für jede offene Dateinummer ist ein separater Pufferbereich vorgesehen. Die Werte werden den Variablen in der Variablenliste von links nach rechts zugeordnet, wobei die Daten in diesem Pufferspeicher verwendet werden. Ist die Variablenliste mit den entsprechenden Werten aufgefüllt, werden alle übrigen Elemente im Pufferspeicher gelöscht, wenn nicht das INPUT—Statement mit einem abschließenden Komma endet. Die Anwendung eines abschließenden Kommas schafft eine "schwebende" Eingabe-Bedingung.

```
>NEW

>100 OPEN #13:"CS1",SEQUENTIA
 L,DISPLAY,INPUT,FIXED 64
>110 INPUT #13:A,B,C,D
 .
 . Programmzeilen
 .
>290 CLOSE #13
>300 END
>RUN

 --1. EINGABE-SATZ =22,77,56,
                    92

 -- Ergebnisse
     A=22   B=77   C=56   D=92
```

Ist die Variablenliste im INPUT-Statement länger als die Anzahl der Datenelemente in der momentanen Aufzeichnung, die nun verarbeitet werden soll, erhält der Computer den nächsten Satz von der Datei und verwendet seine Datenelemente zur Vervollständigung der Variablenliste (siehe Beispiel rechts).

Bei der Durchführung des INPUT-Statements berücksichtigt der Computer, ob die Daten im DISPLAY- oder im INTERNAL-Format gespeichert sind.

```
>NEW

>100 OPEN #13:"CS1",SEQUENTIA
 L,DISPLAY,INPUT,FIXED 64
>110 INPUT #13:A,B,C,D,E,F,G
 .
 . Programmzeilen
 .
>400 END

  --1ST INPUT RECORD=22,33.5
  --2ND INPUT RECORD=405,92
  --3RD INPUT RECORD=22,11023
  --4TH INPUT RECORD=99,100

  --Ergebnisse

    A=22   B=33.5   C=405   D=92

    E=-22  F=11023  G=99
```

INPUT

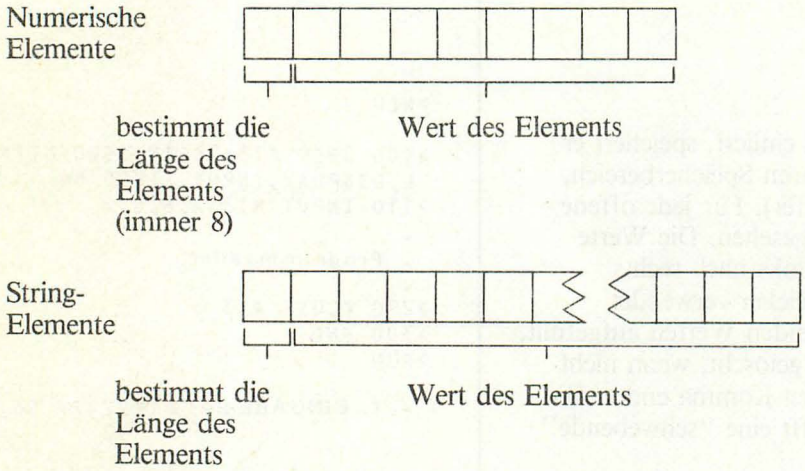
Daten im DISPLAY-Format

Daten im DISPLAY-Format haben die gleiche Form wie die Daten, die über die Tastatur eingegeben werden. Der Computer erkennt die Länge jedes Datenelements in einer DISPLAY-Format-Aufzeichnung an dem Komma-Trennzeichen, das zwischen jedes Element gesetzt ist.

Jedes Element in einer DISPLAY-Format-Aufzeichnung wird kontrolliert, um sicherzustellen, daß numerische Werte für numerische Variable eingesetzt werden, wie rechts in der Aufzeichnung 1 gezeigt wird. Wenn der Datentypus wie rechts in der Aufzeichnung 2 nicht übereinstimmt (JG ist keine numerische Variable), ergibt sich ein Eingabefehler (INPUT ERROR) und das Programm endet.

Daten im INTERNAL-Format

Daten im INTERNAL-Format haben die folgende Form:



Der Computer erkennt die Länge jedes Elements im INTERNAL-Format durch Interpretation des einstelligen Längen-Indikators zu Beginn jedes Elements.

Für Datenelemente im INTERNAL-Format wird eine begrenzte Gültigkeitsprüfung durchgeführt.

Alle numerischen Elemente müssen eine Länge von 9 Stellen haben (8 Stellen plus eine Position für die Längenangabe), und sie müssen gültige Darstellungen von Gleitkommazahlen sein. Andernfalls ergibt sich ein "INPUT ERROR" (Eingabefehler) und das Programm endet.

Bei Aufzeichnungen im INTERNAL-Format mit fester Länge bewirkt das Lesen über die tatsächlichen aufgezeichneten Daten hinaus, daß in jedem Fall Füllzeichen (Binärnullen) gelesen werden. Wenn Sie versuchen, diese Zeichen einer numerischen Variablen zuzuordnen, ergibt sich ein "INPUT ERROR". Wenn Strings gelesen werden sollen, wird der String-Variablen ein Nullstring zugeordnet.

Beispiele:

```
>NEW

>100 OPEN #13:"CS1",SEQUENTIA
 L,DISPLAY,INPUT,FIXED 64
>110 INPUT #13:A,B,STATE$,D$,
 X,Y

 --INPUT RECORD 1=22,97.6,
     TEXAS,"AUTO LICENSE ",
     22000,-.07

 --INPUT RECORD 2=JG,22,TEXAS,
     PROPERTY TAX,42,15
```

INPUT

Die Anwendung von INPUT bei Dateien mit direktem Zugriff (RELATIVE)

(Zur Beschreibung der Dateien mit direktem Zugriff "OPEN")

Dateien mit direktem Zugriff können entweder sequentiell oder mit wahlfreiem Zugriff gelesen werden. Der Computer setzt einen internen Zähler in Gang, der angibt, welche Aufzeichnung als nächste verarbeitet werden soll. Die erste Aufzeichnung in einer Datei ist die Aufzeichnung 0. Der Zähler beginnt also bei 0, und wird bei jedem Zugriff auf die Datei um jeweils +1 erhöht, gleich, ob eine Aufzeichnung gelesen oder geschrieben wird. Im Beispiel rechts weisen die Statements den Computer an, die Datei fortlaufend zu lesen.

Durch Anwendung der REC-Eintragung kann der interne Zähler geändert werden. Der numerische Ausdruck nach dem Schlüsselwort REC wird dazu verwendet, einen bestimmten Satz in der Datei zu identifizieren. Wenn der Computer ein INPUT-Statement mit einer REC-Eintragung durchführt, liest er den angegebenenSatz von der benannten Datei und speichert ihn im E/A-Puffer. Die REC-Eintragung kann nur in Statements verwendet werden, die auf Direktzugriffsdateien verweisen. Das Beispiel rechts veranschaulicht den wahlfreien Zugriff auf eine RELATIVE-Datei über die REC-Eintragung.

Achten Sie darauf, die REC-Eintragung zu verwenden, wenn Sie Sätze aus derselben Datei innerhalb eines Programms schreiben oder lesen. Da derselbe interne Zähler erhöht wird, wenn Sätze für die gleiche Datei gelesen oder geschrieben werden, besteht die Möglichkeit, daß Sie einige Sätze überspringen und andere überschreiben, wenn REC nicht verwendet wird (siehe Beispiel rechts).

Wenn der interne Zähler auf einen Satz außerhalb der Grenzen der Datei verweist, und der Computer den Zugriff auf die Datei versucht, endet das Programm mit einem "INPUT ERROR" (Eingabefehler).

Beispiele:

```
>NEW

>100 OPEN #4:NAME$,RELATIVE,I
 NTERNAL,INPUT,FIXED 64
>110 INPUT #4:A,B,C$,D$,X
 .
 . Programmzeilen
 .
>200 CLOSE #4
>210 END
```

```
>NEW

>100 OPEN #6:NAME$,RELATIVE,I
 NTERNAL,UPDATE,FIXED 72
>110 INPUT K
>120 INPUT #6,REC K:A,B,C$,D$
 .
 . Programmzeilen
 .
>170 PRINT #6,REC K:A,B,C$,D$
 .
 . Programmzeilen
 .
>300 CLOSE #6
>310 END
```

```
>NEW

>100 OPEN #3:NAME$,RELATIVE,I
 NTERNAL,UPDATE,FIXED
>110 FOR I=1 TO 10
>120 INPUT #3:A$,B$,C$,X,Y
 .
 . Programmzeilen
 .
>230 PRINT #3:A$,B$,C$,X,Y
>240 NEXT I
>250 CLOSE #3
>260 END
>RUN

 --Zeile 120-liest die Sätze
              0,2,4,6,8...

 --Zeile 230 -schreibt die Sätze
              1,3,5,7,9...
```

INPUT

Die Anwendung von offenen ("schwebenden") Eingabebedingungen

Eine offene oder schwebende Eingabebedingung ergibt sich dann, wenn ein INPUT-Statement mit einem abschließenden Komma durchgeführt wird. Bei Erreichen des nächsten INPUT-Statements, das diese Datei verwendet, findet einer der folgenden Vorgänge statt:

- Hat das nächste INPUT-Statement keine REC-Eintragung, verwendet der Computer die Daten im E/A-Puffer von der Stelle an, wo das vorangehende INPUT-Statement stoppte.
- Ist dem nächsten INPUT-Statement eine REC-Eintragung beigegeben, beendet der Computer die schwebende Eingabebedingung und liest den spezifizierten Satz in den E/A-Puffer der Datei.

Wenn eine schwebende Eingabebedingung existiert, und man führt für die gleiche Datei ein PRINT-Statement durch, wird die schwebende Eingabebedingung beendet und das PRINT-Statement findet in der üblichen Weise statt.

Bei Anwendung einer schwebenden Eingabebedingung mit Dateinummer 0 wird die Nachricht "INCORRECT STATEMENT" (Anweisung nicht korrekt) ausgedruckt und der Programmablauf unterbrochen.

Datei-Ende

Um bei der seriellen Verarbeitung Fehler zu vermeiden, wenn der Computer keine weiteren Daten mehr zu lesen hat, müssen Sie eine entsprechende Information geben, daß das Ende der Datei erreicht wurde.

Um diese Maßnahme für Sie zu vereinfachen, wurde in das TI-BASIC die Datei-Ende-Funktion (EOF-Funktion) aufgenommen.

Achten Sie darauf, daß Sie das EOF-Statement unmittelbar nach dem INPUT-Statement aufnehmen, das die seriell organisierte Datei liest. Auf diese Weise können Sie den Computer ohne Schwierigkeiten veranlassen, das Lesen der Eingabedatei zu stoppen, wenn keine Daten mehr zur Verfügung stehen. Das gewöhnliche Verfahren ist, zu einer Abschlußroutine zu springen, wenn das EOF-Statement erreicht wurde.

Beispiele:

```
>NEW

>100 INPUT #0:A,B,
>110 PRINT A;B
>120 GOTO 100
>RUN
 ?
 * INCORRECT STATEMENT
    IN 100
```

```
>NEW

>100 OPEN #5:NAME$,SEQUENTIAL
 ,INTERNAL,INPUT,FIXED
>110 IF EOF(5) THEN 150
>120 INPUT #5:A,B
>130 PRINT A;B
>140 GOTO 110
>150 CLOSE #5
>160 END
```

Die EOF-Funktion kann nicht in Verbindung mit Dateien mit direktem Zugriff (RELATIVE) bzw. mit einigen Peripheriegeräten verwendet werden. In diesen Fällen müssen Sie Ihre eigene Methode entwickeln, um festzulegen, wann das Datei-Ende erreicht ist.

Eine allgemeine Datei-Ende-Technik ist die Schaffung einer letzten Aufnahme auf der Datei selbst, die als Indikator für das Ende dient. Man spricht in diesem Zusammenhang von einem Pseudo-Satz, weil die in ihr enthaltenen Daten lediglich das Ende der Datei markieren.

Der Pseudo-Satz könnte zum Beispiel mit einer Ziffernfolge, bestehend aus der Zahl 9, erfolgen. Immer, wenn der Computer einen Satz liest, können Sie die Daten prüfen. Sind die Daten dann gleich 9, hat der Computer das Datei-Ende erreicht, und kann zur Abschlußroutine übergehen.

Das erste Beispiel rechts erzeugt einen Pseudo-Satz. Im nächsten Beispiel kontrolliert der Computer das Datei-Ende-Verfahren über diesen Pseudo-Satz.

Kassettenrekorder-Informationen

- Dateien mit direktem Zugriff können in Verbindung mit Kassettenrekordern nicht verwendet werden.
- Die Anwendung der EOF-Funktion ist bei Dateien auf dem Kassettenrekorder nicht möglich.
- Sie können eine Aufzeichnungslänge bis maximal 192 Stellen spezifizieren.
- Nur Kassettenrekorder 1 (CS1) kann zum Laden von Daten benutzt werden.

Beispiele:

```
>NEW

>100 OPEN #2:"CS1",SEQUENTIAL
 ,FIXED,OUTPUT,INTERNAL
>110 READ A,B,C
>120 IF A=99 THEN 180
>130 E=A+B+C
>140 PRINT A;B;C;E
>150 PRINT #2:A,B,C,E
>160 GOTO 110
>170 DATA 5,10,15,10,20,30,10
 0,200,300,99,99,99
>180 PRINT #2:99,99,99,99
>190 CLOSE #2
>200 END
>RUN

 * REWIND CASSETTE TAPE    CS1
   THEN PRESS ENTER

 * PRESS CASSETTE RECORD   CS1
   THEN PRESS ENTER
  5  10  15  30
  10  20  30  60
  100  200  300  600

 * PRESS CASSETTE STOP     CS1
   THEN PRESS ENTER

 ** DONE **
```

```
>NEW

>100 OPEN #1:"CS1",INTERNAL,I
 NPUT,FIXED
>110 INPUT #1:A,B,C,E
>120 IF A=99 THEN 160
>130 F=A*E
>140 PRINT A;B;C;E;F
>150 GOTO 110
>160 CLOSE #1
>170 END
>RUN

 * REWIND CASSETTE TAPE    CS1
   THEN PRESS ENTER

 * PRESS CASSETTE PLAY     CS1
   THEN PRESS ENTER
  5  10  15  30  150
  10  20  30  60  600
  100  200  300  600  6000

 * PRESS CASSETTE STOP     CS1
   THEN PRESS ENTER

 ** DONE **
```

EOF - Datei-Ende-Funktion

EOF *(numerischer Ausdruck)*

Die Datei-Ende-Funktion bestimmt, wann das Ende einer Datei erreicht wird, die auf einem Peripheriegerät gespeichert ist. Das Argument gibt die Nummer einer offenen Datei an (siehe Seite 123). Das Argument ist der Wert, den Sie erhalten, wenn der numerische Ausdruck berechnet wird. Für die Berechnung von numerischen Argumenten gibt die Nummer eine offene Datei an (siehe "OPEN").

Der Wert aus der Funktion hängt von der Position der Datei ab. Die Werte sind:

Wert	Position
0	Datei-Ende nicht erreicht
+1	Logisches Ende der Datei
-1	tatsächliches Ende der Datei

Eine Datei befindet sich an ihrem logischen Ende, wenn alle ihre Aufzeichnungen verarbeitet worden sind. Das tatsächliche Ende der Datei liegt dann vor, wenn kein weiterer Raum mehr verfügbar ist.

Diese Funktion und das Beispiel rechts können nicht in Verbindung mit Kassettenrekordern verwendet werden. Die Anwendung mit anderen Peripheriegeräten wird in den entsprechenden Bedienungsanleitungen ausführlich erläutert.

Beispiele:

```
>NEW

>100 OPEN #2:NAME$,SEQUENTIAL
 ,INTERNAL,INPUT,FIXED
>110 IF EOF(2) THEN 160
>120 REM IF EOF GIVES ZERO
>130 INPUT #2:A,B,C
>140 PRINT A;B;C
>150 GOTO 110
>160 CLOSE #2
>170 END
```

PRINT

PRINT # *Dateinummer* [,REC *numerischer Ausdruck*] [:*Druckliste*]

Diese Form des PRINT-Statements erlaubt das Schreiben (das Übertragen) von Daten auf ein Peripheriegerät. Das PRINT-Statement kann zum Schreiben auf Dateien ausschließlich dann verwendet werden, wenn diese im OUTPUT-, UPDATE- oder APPEND-Modus eröffnet wurden . Die Dateinummer muß die Nummer einer momentan offenen Datei sein.

Wenn der Computer ein PRINT-Statement durchführt, speichert er die Daten in einem temporären Speicherbereich, dem sogenannten Ein-/Ausgabe-Puffer (E/A-Puffer). Für jede offene Dateinummer ist ein separater Pufferbereich vorgesehen. Wird das PRINT-Statement nicht mit einem Druckseparator (Komma, Semikolon oder Doppelpunkt) beendet, wird der Datensatz vom E/A-Puffer unmittelbar auf die Datei überschrieben. Wenn das PRINT-Statement dagegen mit einem Druckseparator endet, werden die Daten im Puffer gehalten und eine "schwebende" Druckbedingung geschaffen.

Die Informationen, die Sie für die Bildung einer Druckliste benötigen, um die Daten auf Peripheriespeichergeräte aufzuzeichnen, werden nachstehend angegeben. Die erforderliche Druckliste zur Anzeige von Druckzeilen (auf einem Drucker etc.) ist im Abschnitt „INPUT-OUTPUT-Statements" beschrieben. Für die auf den Peripheriegeräten gespeicherten Daten können Sie entweder das DISPLAY- oder das INTERNAL-Format verwenden. Da diese Dateien jedoch nur vom Computer gelesen werden, ist der bei weitem einfachste und effektivste Datentyp das INTERNAL-Format.

Die Anwendung von PRINT mit Daten im INTERNAL-Format

Die Druckliste besteht aus numerischen und String- Ausdrücken, die durch Kommas, Doppelpunkte oder durch das Semikolon getrennt sind. Alle Druckseparatoren in einer Druckliste haben die gleiche Wirkung für Daten im INTERNAL-Format - sie trennen nur die einzelnen Elemente voneinander und indizieren keine Leerzeichenpositionen in einem Datensatz.

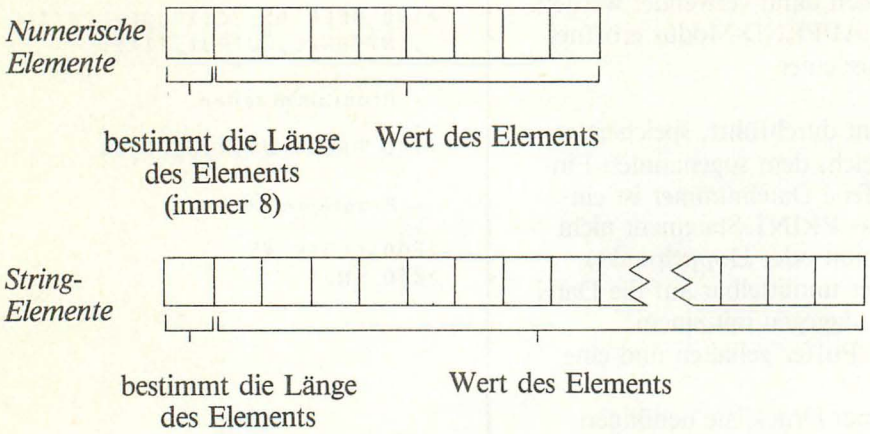
Beispiele:

```
>NEW

>100 OPEN #5:"CS1",SEQUENTIAL
 ,INTERNAL,OUTPUT,FIXED
 .
 . Programmzeilen
 .
>170 PRINT #5:A,B,C$,D$
 .
 . Programmzeilen
 .
>200 CLOSE #5
>210 END
```

```
>NEW

>100 OPEN #6:"CS2",SEQUENTIAL
 ,DISPLAY,OUTPUT,FIXED
 .
 . Programmzeilen
 .
>170 PRINT #6:A;",";B;",";C$;
 ",";D$
 .
 . Programmzeilen
 .
>200 CLOSE #6
>210 END
```

PRINT

Wenn die Elemente in der Druckliste im INTERNAL-Format auf ein Peripheriegerät überschrieben werden, haben sie folgende Eigenschaften:



Im Beispiel rechts beträgt die Gesamtlänge der im INTERNAL-Format aufgezeichneten Daten 71 Stellen. Jede numerische Variable verwendet 9 Stellen. A$ ist ein 18 Zeichen umfassender String (Zeile 110) plus eine Stelle zur Aufzeichnung der Stringlänge. B$ beträgt 15 Zeichen plus 1. Wenn sich die Werte von A$ und B$ während des Programms ändern, variieren ihre Längen je nachdem, welchen Wert sie gerade haben, wenn der Datensatz auf die Dateien überschrieben wird. Berücksichtigen Sie daher bei der Planung Ihres Datensatzes die Werte, die jede Variable enthalten könnte, und berechnen Sie für die Satzlänge die größtmögliche Länge.

Immer, wenn Sie Datensätze mit fester Länge spezifizieren, füllt der Computer jede Datensatz im INTERNAL-Format bei Bedarf mit Binärnullen auf, um jeden Datensatz auf die gleiche angegebene Länge zu bringen.

Der Computer erlaubt keine Datensätze, die länger sind als angegeben, oder die die Standardlänge des verwendeten Peripheriegerätes überschreiten. Wenn die Aufnahme aller Daten in die Druckliste diese Bedingung für einen Datensatz im INTERNAL-Format zur Folge hat, endet das Programm mit der Nachricht "FILE ERROR in xx".

Beispiele:

```
>NEW

>100 OPEN #5:"CS1",SEQUENTIAL
 ,INTERNAL,OUTPUT,FIXED 128
>110 A$="TEXAS INSTRUMENTS "
>120 B$="HOME COMPUTER "
>130 READ X,Y,Z
>140 IF X=99 THEN 190
>150 A=X*Y*Z
>160 PRINT #5:A$,X,Y,Z,B$,A
>170 GOTO 130
>180 DATA 5,6,7,1,2,3,10,20,3
 0,20,40,60,1.5,2.3,7.6,99,99
 ,99
>190 CLOSE #5
>200 END
>RUN

 * REWIND CASSETTE TAPE    CS1
   THEN PRESS ENTER

 * PRESS CASSETTE RECORD   CS1
   THEN PRESS ENTER

 -- Daten auf Kassette
    gespeichert

 * PRESS CASSETTE STOP     CS1
   THEN PRESS ENTER

 ** DONE **
```

Die Anwendung von PRINT mit Daten im DISPLAY-Format auf Datei-Speichereinheiten

Obwohl man für Daten, die vorzugsweise der Computer liest, das INTERNAL-Format verwenden sollte, gibt es gelegentlich Situationen, in denen man die Datensätze im DISPLAY-Format benötigt.

Dieses Kapitel enthält einige wichtige Überlegungen, die bei der Anwendung des DISPLAY-Formats zu berücksichtigen sind.

• Die Sätze werden entsprechend den Spezifikationen im PRINT-Statement im Abschnitt „Ein-/Ausgabe-Anweisungen gebildet.

• Wenn die Einbeziehung eines Datenelements aus der Druckliste bewirken würde, daß der Datensatz länger als angegeben wird, oder daß die Standardlänge das verwendeten Geräts überschritten wird, erfolgt keine Aufsplittung des Elements, sondern es wird das erste Element im nächsten Datensatz. Ist ein Einzelelement länger als die mögliche Satzlänge, wird es in so viele Sätze aufgeteilt, wie zur Speicherung erforderlich sind. Der Programmablauf wird ohne Fehlermeldung in der üblichen Form weitergeführt.

• Um die Dateien im Display-Format, die mit dem PRINT-Statement gebildet wurden, später zu lesen, müssen die Daten so aussehen, wie bei der Eingabe über die Tastatur. Es müssen also die im INPUT-Statement nötigen Komma-Separatoren und Anführungszeichen ausdrücklich eingefügt werden, wenn der Datensatz auf die Datei überschrieben wird. Diese Interpunktionszeichen werden bei der Ausführung des PRINT-Statements nicht automatisch eingefügt. Sie müssen als Elemente mit in die Druckliste aufgenommen werden, wie in Zeile 170 rechts gezeigt.

• Numerische Elemente haben keine feste Länge wie im INTERNAL-Format. Bei Dateien im DISPLAY-Format ist die Länge eines numerischen Elements genauso, wie sie auf dem Bildschirm nach Anwendung des PRINT-Statements oder des DISPLAY-Statements angezeigt wird (d. h., sie umfaßt das Vorzeichen, den Dezimalpunkt, den Exponenten, abschließende Leerstellen usw.). Zum Beispiel sind für das Ausdrucken von 1.35E-10 insgesamt 10 Stellen erforderlich.

Beispiele:

```
>NEW

>100 OPEN #10:"CS1",SEQUENTIA
 L,DISPLAY,OUTPUT,FIXED 128
 .
 . Programmzeilen
 .
>170 PRINT #10:"""";A$;"""",";
 X;",";Y;",";Z;",""";B$;""","
 ;A
 .
 . Programmzeilen
 .
>300 CLOSE #10
>310 END
```

PRINT

Die Anwendung von PRINT bei Dateien mit direktem Zugriff
(Eine Beschreibung der Dateiorganisation mit wahlfreiem Zugriff finden Sie unter „OPEN".)

Sätze von Dateien mit direktem Zugriff können in beliebiger Folge oder nacheinander verarbeitet werden. Der Computer verwendet einen internen Zeiger, um zu markieren, welcher Datensatz als nächster zu verarbeiten ist. Die erste Aufzeichnung in einer Datei ist der Satz 0. Der Zähler beginnt also bei 0 und wird nach jedem Zugriff auf die Datei um +1 erhöht, gleich, ob der Datensatz gelesen oder geschrieben wird. Im Beispiel rechts weisen die Statements den Computer an, die Datei fortlaufend zu lesen. Später kann die Verarbeitung sequentiell oder in beliebiger Reihenfolge stattfinden.

Beispiele:

```
>NEW

>100 OPEN #3:NAME$,RELATIVE,I
 NTERNAL,OUTPUT,FIXED 128
 .
 . Programmzeilen
 .
>150 PRINT #3:A$,B$,C$,X,Y,Z
 .
 . Programmzeilen
 .
>200 CLOSE #3
>210 END
```

Durch Anwendung der REC-Eintragung kann der interne Zähler geändert werden. Dem Schlüsselwort REC muß ein numerischer Ausdruck folgen, dessen Wert angibt, an welche Stelle der Datensatz zu überschreiben ist. Wenn der Computer ein PRINT-Statement mit einer REC-Eintragung ausführt, bildet er zunächst einen Ausgabesatz im E/A-Puffer. Ist dieser Satz dann auf die Datei überschrieben, wird er an die Stelle gebracht, die durch die REC-Eintragung spezifiziert wurde. Die REC-Eintragung darf nur in Dateien mit direktem Zugriff verwendet werden. Das Beispiel rechts veranschaulicht das Überschreiben von Datensätzen in beliebiger Folge, wobei die REC-Eintragung angewandt wird.

```
>NEW

>100 OPEN #3:NAME$,RELATIVE,I
 NTERNAL,UPDATE,FIXED 128
>110 INPUT K
>120 INPUT #3,REC K:A$,B$,C$,
 X,Y,Z
 .
 . Programmzeilen
 .
>300 CLOSE #3
>310 END
```

Achten Sie darauf, die REC-Eintragung zu verwenden, wenn Sie Datensätze der gleichen Datei innerhalb eines Programms lesen und schreiben. Da derselbe interne Zähler betätigt wird, wenn Datensätze für die gleiche Datei gelesen oder geschrieben werden, besteht die Gefahr, daß Sie einige Sätze überspringen und andere überschreiben, wenn REC nicht verwendet wird (siehe Beispiel rechts).

```
>NEW

>100 OPEN #3:NAME$,RELATIVE,I
 NTERNAL,UPDATE,FIXED
>110 FOR I=1 TO 10
>120 PRINT #3:A$,B$,C$,X,Y
>130 PRINT #3:A$,B$,C$,X,Y
>140 NEXT I
>150 CLOSE #3
>160 END

Zeile 120 -schreibt die Sätze 0,2,4,
                              6,8...

Zeile 130 -schreibt die Sätze 1,3,
                              5,7,
                              9...
```

Die Anwendung von offenen ("schwebenden") PRINT-Bedingungen

Ein Datensatz wird immer auf eine Datei überschrieben, wenn der Computer ein PRINT-Statement ausführt, das kein abschließendes Trennzeichen (Separator) aufweist. Eine offene PRINT-Bedingung wird bei Ausführung eines PRINT-Statements mit abschließendem Druckseparator geschaffen. Bei Erreichen des nächsten PRINT-Statements, das die Datei verwendet, findet einer der folgenden Vorgänge statt:

- Hat das nächste PRINT-Statement keine REC-Eintragung, plaziert der Computer die Daten im E/A-Puffer unmittelbar nach den bereits vorhandenen Daten.
- Hat das nächste PRINT-Statement eine REC-Eintragung, schreibt der Computer den "schwebenden" Drucksatz auf die Datei in die durch den internen Zähler angegebene Position und führt das nächste PRINT-REC-Statement wie üblich durch.

PRINT

Existiert eine "schwebende" PRINT-Bedingung, und wird für die gleiche Datei ein INPUT-Statement erreicht, wird der "schwebende" (offene) Satz auf die Datei an die Position überschrieben, die der interne Zähler angibt, und dieser Zähler wird anschließend erhöht. Dann wird das INPUT-Statement in der üblichen Weise durchgeführt. Wenn eine schwebende PRINT-Bedingung existiert, und die Datei wird abgeschlossen oder mit einem RESTORE-Statement beeinflußt, wird der schwebende Satz vor der Durchführung von CLOSE oder RESTORE noch überschrieben.

Kassettenrekorder-Informationen

- Sie können jede Aufzeichnungslänge bis zu 192 Stellen spezifizieren
- Nur die Verarbeitung von seriell organisierten Dateien ist möglich.

RESTORE

RESTORE # *Dateinummer* [,*REC numerischer Ausdruck*]
(Eine Beschreibung des RESTORE-Statements in Verbindung mit READ und DATA finden Sie im Abschnitt "INPUT-OUTPUT-Statements".)

Das RESTORE-Statement bringt eine offene Datei wieder an den Anfangs-Datensatz (Anfangsaufzeichnung), oder an einen anderen spezifischen Datensatz, wenn es sich um eine Datei mit direktem Zugriff handelt (siehe hierzu die beiden Beispiele rechts).

Wenn die Datei mit der im RESTORE-Statement spezifizierten Nummer noch nicht offen ist, endet das Programm mit der Nachricht "FILE ERROR IN xx" (Dateifehler in xx).

Die REC-Eintragung darf nur in Verbindung mit Dateien mit direktem Zugriff verwendet werden. Der Computer wertet den numerischen Ausdruck nach REC aus, und verwendet das Resultat als Zeiger auf einen spezifischen Datensatz (bzw. Aufzeichnung) in der Datei. Wenn Sie RESTORE für eine Datei mit direktem Zugriff ohne REC-Eintragung verwenden, wird die Datei auf den Datensatz 0 gesetzt.

Existiert ein schwebender PRINT-Satz, wird vor der Durchführung von RESTORE der Datensatz noch auf die Datei überschrieben. Bei Existieren einer schwebenden Eingabebedingung werden die Daten im E/A-Puffer gelöscht.

Die Anwendung von Dateien mit direktem Zugriff ist bei Kassettenrekordern nicht möglich.

Beispiele:

```
>NEW

>100 OPEN #2:"CS1",SEQUENTIAL
 ,INTERNAL,INPUT,FIXED 64
>110 INPUT #2:A,B,C$,D$,X
 .
 . Programmzeilen
 .
>400 RESTORE #2
>410 INPUT #2:A,B,C$,D$,X
 .
 . Programmzeilen
 .
>500 CLOSE #2
>510 END


>NEW

>100 OPEN #4:NAME$,RELATIVE,I
 NTERNAL,UPDATE,FIXED 128
>110 INPUT #4:A,B,C
 .
 . Programmzeilen
 .
>200 PRINT #4:A,B,C
 .
 . Programmzeilen
 .
>300 RESTORE #4,REC 10
>310 INPUT #4:A,B,C
 .
 . Programmzeilen
 .
>400 CLOSE #4
>410 END
```

Fehlermeldungen

I. Fehler bei Eingabe einer Zeile

*BAD LINE NUMBER (falsche Zeilennummer)

1. Die Zeilennummer oder die Bezugnahme auf eine Zeilennummer = 0 oder > 32767.
2. Die RESEQUENCE-Spezifikationen erzeugen eine Zeilennummer > 32767.

*BAD NAME (falsche Bezeichnung)

1. Der Variablenname hat mehr als 15 Zeichen.

*CAN'T CONTINUE (Fortsetzung nicht möglich)

1. Der CONTINUE-Befehl wurde ohne vorangehende Stoppstelle eingegeben, oder das Programm wurde seit der Stoppstelle schon editiert.

*CAN'T DO THAT (Durchführung nicht möglich)

1. Versuch, die folgenden Programmanweisungen (Statements) als Befehle zu verwenden: DATA, DEF, FOR, GOTO, GOSUB, IF, INPUT, NEXT, ON, OPTION, RETURN.
2. Versuch, die folgenden Befehle als Programm-Statements (mit Eingabe einer Zeilennummer) zu verwenden: BYE, CONTINUE, EDIT, LIST, NEW, NUMBER, OLD, RUN, SAVE.
3. Eingabe von LIST, RUN oder SAVE ohne Programm.

*INCORRECT STATEMENT (Statement nicht korrekt)

1. Zwei Variablenbezeichnungen in einer Reihe ohne dazwischenliegendes gültiges Trennzeichen (ABC A oder A$A).
2. Eine numerische Konstante folgt unmittelbar nach einer Variablen ohne dazwischenliegendes gültiges Trennzeichen (N 257).
3. Ein in Anführungszeichen begonnener String hat keine abschließenden Anführungszeichen.
4. Ungültiger Druckseparator (Trennzeichen) zwischen den Zahlen bei den Befehlen LIST, NUMBER oder RESEQUENCE.
5. Ungültige Zeichen nach den Befehlen CONTINUE, LIST, NUMBER, RESEQUENCE oder RUN.
6. Das Befehls-Schlüsselwort ist nicht das erste Wort in der Zeile.
7. Der Doppelpunkt nach der Gerätebezeichnung in einem LIST-Befehl fehlt.

*LINE TOO LONG (Zeile zu lang)

1. Die Eingabezeile ist zu lang für den Eingabe-Pufferbereich.

*MEMORY FULL (Speicher belegt)

1. Eingabe einer Editierzeile, die den verfügbaren Speicherplatz überschreitet.
2. Das Addieren einer Zeile zum Programm bewirkt, daß das Programm den verfügbaren Speicherbereich überschreitet.

II. Fehler bei der Aufstellung der Symboltabelle

Gibt man RUN ein, noch ehe Programmzeilen durchgeführt sind, tastet der Computer das Programm ab, um eine Symboltabelle zu bilden. Eine Symboltabelle ist der Bereich des Speichers, wo die Variablen, Datenfelder, Funktionen usw. für ein Programm gespeichert werden. Während dieses Abtastverfahrens erkennt der Computer Programmfehler. Die Nummer der Zeile, die den Fehler enthält, wird als Teil der Meldung mit ausgedruckt (zum Beispiel: *BAD VALUE IN 100 – falscher Wert in Zeile 100). Da an dieser Stelle noch keine Programmzeilen durchgeführt wurden, sind alle Werte in der Symboltabelle null.

*BAD VALUE IN 100 - falscher Wert in Zeile 100). Da an dieser Stelle noch keine Programmzeilen durchgeführt wurden, sind alle Werte in der Symboltabelle null.

*BAD VALUE (falscher Wert)

1. Eine Dimension für ein Datenfeld ist größer als 32767.
2. Eine Dimension für ein Datenfeld ist null, wenn OPTION BASE = 1.

*CAN'T DO THAT (Durchführung nicht möglich)

1. Mehr als ein OPTION BASE Statement im Programm.
2. Das OPTION BASE Statement hat eine höhere Zeilennummer als eine Datenfeld-Definition.

*FOR-NEXT ERROR (FOR-NEXT-Fehler)

1. Unvereinbarkeit der Zahlen von FOR- und NEXT-Statements.

*INCORRECT STATEMENT (Statement nicht korrekt)

DEF

1. Keine abschließende Klammer "(" nach einem Parameter in einem DEF-Statement.
2. Das Gleichheitszeichen (=) fehlt im DEF-Statement.
3. Der Parameter im DEF-Statement ist kein gültiger Variablenname.

Fehlermeldungen

DIM

4. Das DIM-Statement hat keine oder mehr als drei Dimensionen.
5. Eine Dimension im DIM-Statement ist keine Zahl.
6. Einer Dimension im DIM-Statement folgt kein Komma oder keine abschließende Klammer "(".
7. Die Datenfeld-Bezeichnung in einem DIM-Statement ist kein gültiger Variablenname.
8. Die abschließende Klammer "(" fehlt für Datenfeld-Indices.

OPTION BASE

9. Das Wort BASE folgt nicht nach dem Wort OPTION.
10. Nach OPTION BASE fehlt die Zahl 0 oder 1.

*MEMORY FULL (Speicher belegt)

1. Das Datenfeld ist zu umfangreich.
2. Der Speicherplatz ist nicht ausreichend, um eine Variable oder Funktion zuzuordnen.

*NAME CONFLICT (Widerspruch in den Bezeichnungen)

1. Der gleiche Name wird mehr als einem Datenfeld zugeordnet (DIM A (5), A (2,7)).
2. Der gleiche Name wird einem Datenfeld und einer einfachen Variablen zugeordnet.
3. Der gleiche Name wird einer Variablen und einer Funktion zugeordnet.
4. Verweisungen auf ein Datenfeld geben eine unterschiedliche Anzahl von Dimensionen für das Datenfeld an (B.-. A (2,7) + 2.PRINT A (5)).

III. Fehler beim Ablauf eines Programms

Beim Ablauf eines Programms besteht die Möglichkeit, daß der Computer auf Statements trifft, die er nicht durchführen kann. Eine Fehlermeldung wird ausgedruckt, und wenn der Fehler nicht nur eine Warnung ist, endet das Programm. An dieser Stelle haben alle Variablen im Programm die Werte, die bei Auftreten des Fehlers zugeordnet waren. Die Nummer der Zeile, in der der Fehler enthalten ist, wird als Teil der Nachricht mit angegeben (zum Beispiel: CAN'T DO THAT IN 210).

*BAD ARGUMENT (falsches Argument)

1. Eine eingebaute Funktion hat ein falsches Argument.
2. Der String-Ausdruck für die eingebauten Funktionen ASC oder VAL hat die Länge null (Null-String).
3. In der VAL-Funktion ist der String-Ausdruck keine gültige Darstellung einer numerischen Konstanten.

*BAD LINE NUMBER (falsche Zeilennummer)

1. Die angegebene Zeilennummer existiert nicht im ON-, GOTO- oder GOSUB-Statement.
2. Die in BREAK oder UNBREAK spezifizierte Zeilennummer existiert nicht (nur Warnung).

*BAD NAME (falsche Bezeichnung)

1. Der Unterprogramm-Name in einem CALL-Statement ist ungültig.

*BAD SUBSCRIPT (falscher Index)

1. Der Index ist keine ganze Zahl.
2. Der Index hat einen größeren Wert als angegeben oder überschreitet die erlaubten Dimensionen eines Datenfelds.
3. Der Index 0 wird verwendet, wenn OPTION BASE 1 spezifiziert wurde.

*BAD VALUE (falscher Wert)

CHAR

1. Der Zeichenkode liegt außerhalb des Bereichs im CHAR-Statement.
2. Das Zeichen beim Muster-Identifizierer im CHAR-Statement ist ungültig.

CHR$

3. Das Argument ist negativ oder länger als +32767 bei CHR$ COLOR.
4. Die Zeichensatznummer ist außerhalb des Bereichs im COLOR-Statement.
5. Der Farbkode für Vorder- oder Hintergrund liegt nicht im Bereich des COLOR-Statements.

POTENZIERUNG

6. Versuch, eine negative Zahl in eine gebrochene Potenz zu erheben.

FOR

7. Das Schritt-Inkrement ist null im FOR-TO-STEP-Statement.

HCHAR, VCHAR, GCHAR

8. Die Reihen- oder Spaltennummer liegt nicht im Bereich des HCHAR-, VCHAR- oder GCHAR-Statements.

JOYST, KEY

9. Die Tasteneinheit liegt nicht im Bereich des JOYST-oder KEY-Statements.

ON

10. Der numerische Ausdruck, der die Zeilennummer indiziert, liegt nicht im Definitionsbereich.

Fehlermeldungen

OPEN, CLOSE, INPUT, PRINT, RESTORE

11. Die Dateinummer ist negativ oder größer als 255.
12. Die Zahl der Datensätze (Aufzeichnung) in der SEQUENTIAL-Alternative des OPEN-Statements ist nicht-numerisch oder größer als 32767.
13. Die Satzlänge in der FIXED-Alternative des OPEN-Statements ist größer als 32767.

POS

14. Der numerische Ausdruck im POS-Statement ist negativ, null oder größer als 32767.

SCREEN

15. Der Bildschirm-Farbkode liegt nicht im Definitionsbereich.

SEG$

16. Der Wert des numerischen Ausdrucks 1 (Zeichenposition) oder des numerischen Ausdrucks 2 (Länge des SUB-Strings) ist negativ oder größer als 32767.

SOUND

17. Dauer-, Frequenz-, Volumen- oder Rauschspezifikation liegen nicht im angegebenen Bereich.

TAB

18. Der Wert der Zeichenposition in der Spezifikation der TAB-Funktion ist größer als 32767.

*CAN'T DO THAT (Durchführung nicht möglich)

1. RETURN ohne vorangehendes GOSUB-Statement.
2. NEXT ohne vorangestelltes paariges FOR-Statement.
3. Die Kontrollvariable im NEXT-Statement paßt nicht zur Kontrollvariablen im vorangehenden FOR-Statement.
4. BREAK-Befehl ohne Zeilennummer.

*DATA ERROR (Datenfehler)

1. Kein Komma zwischen den Elementen im DATA-Statement.
2. Die Variablen-Liste im READ-Statement ist nicht aufgefüllt, aber weitere DATA-Statements sind nicht verfügbar.
3. READ-Statement ohne verbleibendes DATA-Statement.
4. Zuordnung eines String-Wertes zu einer numerischen Variablen in einem READ-Statement.
5. Zeilennummer im RESTORE-Statement ist größer als die höchste Zeilennummer im Programm.

*FILE ERROR (Dateifehler)

1. Versuch, für eine momentan nicht offene Datei CLOSE, INPUT, PRINT oder RESTORE durchzuführen.
2. Versuch, für Datensätze (Aufzeichnungen) aus einer Datei INPUT durchzuführen, die als OUTPUT oder APPEND geöffnet wurde.
3. Versuch, für Datensätze auf einer Datei PRINT durchzuführen, die als INPUT geöffnet wurde.
4. Versuch, für eine Datei OPEN durchzuführen, die bereits geöffnet ist.

*INCORRECT STATEMENT (Statement nicht korrekt)

Allgemein

1. Einleitende Klammer "(" oder abschließende Klammer ")" oder beide fehlen.
2. Komma fehlt.
3. Die Zeilennummer fehlt dort, wo sie bei BREAK, UNBREAK oder RESTORE erwartet wird (BREAK 100).
4. Auf "+" oder "—" folgt kein numerischer Ausdruck.
5. Ausdrücke, die in Verbindung mit arithmetischen Operatoren verwendet werden, sind nicht numerisch.
6. Ausdrücke, die in Verbindung mit Vergleichsoperatoren verwendet werden, sind nicht vom gleichen Typ.
7. Versuch, einen String-Ausdruck als Index zu verwenden.
8. Versuch, einer Funktion einen Wert zuzuordnen.
9. Reserviertes Wort an unpassender Stelle.
10. Unerwarteter arithmetischer oder Vergleichs-Operator.
11. Erwarteter arithmetischer oder Vergleichs-Operator fehlt.

Eingebaute Unterprogramme

12. Bei JOYST sind x-Rückgabe und y-Rückgabe keine numerischen Variablen.
13. Bei KEY ist der Tastenstatus keine numerische Variable.
14. In GCHAR muß die dritte Spezifikation eine numerische Variable sein.
15. Mehr als drei Ton-Spezifikationen oder mehr als eine Rausch-Spezifikationen in SOUND.
16. Auf CALL folgt keine Unterprogramm-Bezeichnung.

Fehlermeldungen

Datei-Verarbeitung, Ein-/Ausgabe-Statements

17. Nummernzeichen (#) oder Doppelpunkt in der Dateinummer-Spezifikationen für OPEN, CLOSE, INPUT, PRINT oder RESTORE fehlt.
18. Der Dateiname in OPEN oder DELETE muß ein String-Ausdruck sein.
19. Ein Schlüsselwort im OPEN-Statement ist ungültig oder erscheint mehr als einmal.
20. Die Anzahl der Datensätze (Aufzeichnungen) in der SEQUENTIAL-Option ist im OPEN-Statement kleiner als null.
21. Die Satzlänge in der FIXED-Option des OPEN-Statements ist kleiner als null oder größer als 255.
22. Auf einen Doppelpunkt im CLOSE-Statement folgt nicht das Schlüsselwort DELETE.
23. In den PRINT- oder INPUT-Statements fehlt der Doppelpunkt (:), der auf die Dateinummer-Spezifikationen folgen muß (oder REC fehlt, obwohl numerischer Ausdruck darin enthalten).
24. Der Druckseparator (Komma, Doppelpunkt, Semikolon) fehlt im PRINT-Statement dort, wo er erforderlich ist.
25. Der Eingabe-Dialog im INPUT-Statement ist kein String-Ausdruck.
26. Der Dateiname im SAVE-, LOAD- oder OLD-Befehl ist kein gültiger String-Ausdruck.

Allgemeine Programm-Statements (Anweisungen)

FOR

27. Dem Schlüsselwort FOR folgt keine numerische Variable.
28. Im FOR-Statement folgt der Kontroll-Variablen kein Gleichheitszeichen (=).
29. Das Schlüsselwort TO fehlt im FOR-Statement.
30. Im FOR-Statement folgt auf die Grenze nicht das Ende der Zeile oder das Schlüsselwort STEP.

IF

31. Das Schlüsselwort THEN fehlt oder die Zeilennummer nach dem Schlüsselwort THEN fehlt.

LET

32. Das Gleichheitszeichen (=) fehlt im LET-Statement.

NEXT

33. Dem Schlüsselwort NEXT folgt keine Kontroll-Variable.
34. ON-GOTO, ON-GOSUB.
    Nach ON folgt kein gültiger numerischer Ausdruck.

RETURN

35. Nach dem Wort RETURN folgt ein unerwartetes Wort oder Zeichen.

Benutzerdefinierte Funktionen

36. Die Zahl der Funktionsargumente entspricht nicht der Zahl der Parameter für eine benutzerdefinierte Funktion.

*INPUT ERROR (Eingabefehler)

1. Die Eingabedaten sind für den Ein-/Ausgabe-Puffer zu lang (wenn die Daten über die Tastatur eingegeben werden, ist dies nur eine Warnung - eine erneute Eingabe der Daten ist möglich.
2. Die Zahl der Variablen in der Variablenliste entspricht nicht der Eingabezahl von Datenelementen über die Tastatur oder über eine Datei (Warnung nur bei Eingabe über die Tastatur).
3. Nicht-numerische Daten bei der Durchführung von INPUT für eine numerische Variable. Diese Bedingung könnte durch das Lesen von Füllzeichen auf einer Datei-Aufzeichnung verursacht werden. (Warnung nur bei Eingabe über die Tastatur).
4. Numerische INPUT-Daten erzeugen einen Überlauf (Warnung nur bei Eingabe über die Tastatur).

*I/O ERROR (Ein-/Ausgabe-Fehler) - Unter diese Bedingung wird ein Fehlerkode wie folgt erzeugt:

Im Falle eines I/O-Fehlers wird ein zweistelliger Fehlerkode (XY) mit der Nachricht
*I/O ERROR XY IN LINE NUMBER
angezeigt.
Die erste Stelle (X) gibt an, welche I/O-Operation den Fehler verursachte.

X-Wert	Operation
O	OPEN
1	CLOSE
2	INPUT
3	PRINT
4	RESTORE
5	OLD
6	SAVE
7	DELETE

Die zweite Stelle (Y) verweist auf die Art des Fehlers.

Y-Wert	Fehlertyp
0	Gerätebezeichnung wird nichtgefunden (ungültige Geräte- oder Detailbezeichnung im DELETE-LIST-, OLD- oder SAVE Befehl.
1	Geschütztes Gerät (Versuch, eine geschützte Datei zu überschreiben)
2	Falsches OPEN-Attribut (eine oder mehrere Optionen im OPEN-Statement sind unzulässig oder entsprechen nicht den Datei-Eigenschaften)
3	Unzulässige Operation (Ein-/Ausgabe-Anweisung nicht zulässig)
4	Speicherbereichsüberschreitung (Versuch, Daten- oder Programm-Material einzuschreiben, wenn der Platz im Speichermedium nicht mehr ausreicht)
5	Datei-Ende (Versuch, über das Dateiende hinauszulesen)
6	Gerätefehler (Gerät nicht angeschlossen oder defekt; dieser Fehler tritt dann auf, wenn während der Dateiverarbeitung ein Peripheriegerät versehentlich vom System getrennt wird)
7	Dateifehler (die angesprochene Datei existiert nicht oder der Dateityp (Programm oder Daten) wird nicht richtig angesprochen)

*MEMORY FULL (Speicher belegt)

1. Ungenügender Speicherraum zur Zuordnung des spezifizierten Zeichens im CHAR-Statement.
2. Das GOSUB-Statement verzweigt zu seiner eigenen Zeilennummer.
3. Das Programm enthält zu viele schwebende Unterprogrammverzweigungen, ohne daß bisher ein RETURN durchgeführt wurde.
4. Das Programm enthält zu viele benutzerdefinierte Funktionen, die sich wiederum auf andere benutzerdefinierte Funktionen beziehen.
5. Der Vergleichs-, String- oder numerische Ausdruck ist zu lang.
6. Versuch, einer String-Variablen eine Zahl zuzuordnen

*NUMBER TOO BIG (zu große Zahl)
In diesem Fall wird eine Warnung gegeben und der Wert durch die Computergrenze wie folgt ersetzt:

1. Eine numerische Operation erzeugt einen Überlauf (der Wert ist größer als 9.9999999999999E127 oder kleiner als —9.9999999999999E127).
2. Das Lesen aus dem DATA-Statement (READ) führt zu einer Überlaufzuordnung zu einer numerischen Variablen.
3. INPUT führt zu einer Überlaufzuordnung zu einer numerischen Variablen.

*STRING-NUMBER MISMATCH (Stringzahl-Fehlzuordnung)

1. Ein nicht-numerisches Argument wird für eine eingebaute Funktion, eine Tab-Funktion oder eine Potenzierungs-Operation angegeben.
2. In einer Spezifikation, die einen numerischen Wert erfordert, erscheint ein nicht-numerischer Wert.
3. In einer Spezifikation, die einen String-Wert erfordert, erscheint kein String-Wert.
4. Funktionsargument und Parameter stimmen im Typ nicht überein, oder Funktionstyp und Ausdruck sind bei einer benutzerdefinierten Funktion widersprüchlich.
5. Die Dateinummer ist bei OPEN, CLOSE, INPUT, PRINT oder RESTORE nicht numerisch.
6. Versuch, einer Zahl eine String-Variable zuzuordnen

Anmerkung: Weitere Fehlerkodes können bei Anwendung der verschiedenen Peripheriegeräte wie Disketten-System oder Thermodrucker angezeigt werden. Informationen zu diesen Fehlerkodes entnehmen Sie bitte den jeweiligen Bedienungsanleitungen.

IV. Fehlermeldung bei nicht korrekt durchgeführtem OLD-Befehl.
CHECK PROGRAMM IN MEMORY
(gespeichertes Programm prüfen)

Der OLD-Befehl löscht nur dann den Programmspeicher, wenn der Ladevorgang erfolgreich abgeschlossen wurde. Wird dagegen der OLD-Befehl nicht korrekt durchgeführt oder unterbrochen, kann das augenblicklich gespeicherte Programm ganz oder teilweise von dem zu ladenden Programm überschrieben werden. Listen Sie das gespeicherte Programm auf (LIST), ehe Sie fortfahren.

Anwenderprogramme

Einführung
Die Programme in diesem Abschnitt sollen die Anwendung vieler Statements in TI-BASIC veranschaulichen. Wenn Sie noch keinerlei Programmiererfahrung haben, beginnen Sie am besten mit dem Buch "BASIC für Anfänger", im Lieferumfang enthalten. Nach Beendigung der Lektüre und nach dem Durcharbeiten der Programme dort erhalten Sie mit den folgenden Programmbeispielen zusätzliche Unterstützung für eine komplexere Programmierung.

Wenn Sie schon einige Erfahrung besitzen, geben Ihnen die nachstehenden Programme eine Demonstration für viele Merkmale des TI-BASIC.

Sie beginnen auf einfachem Niveau und nehmen allmählich in ihrer Komplexität zu. Sie können also bei jedem gewünschten Niveau beginnen. Bei den meisten Programmen finden auch die Farb- und Akustik-Möglichkeiten des Computers Anwendung. Damit sollen Sie eine solide Basis für die Entwicklung eigener Graphiken erhalten und Ihren Programmen akustische Signale beigeben können.

Die Raupe

Das Programm erzeugt eine Raupe, die sich vorwärts und rückwärts auf dem Bildschirm bewegt. Erreicht sie den Bildschirmrand, ertönt ein "uh-oh" und die Raupe dreht sich um und kriecht in die andere Richtung.

Diese Statements erlauben Ihnen die Eingabe einer Farbe für die Raupe (als Farbkodes werden 2-3, 5-16 empfohlen). Der Bildschirm wird dann gelöscht. Das CALL COLOR Statement ordnet die von Ihnen ausgewählte Farbe dem Zeichensatz 2 zu. XDIR wird verwendet, um die Bewegungsrichtung der Raupe zu bestimmen (+1 bedeutet rechts und —1 links).

Diese Schleife bewegt die Raupe über den Bildschirm. Zeile 180 berechnet, wo der nächste Block anzuzeigen ist und Zeile 190 bringt den neuen Block auf den Bildschirm. Die DELAY-Schleife regelt, wie schnell sich die Raupe über den Bildschirm bewegt. Zeile 220 löscht den alten Farbblock (aus diesem Grund wird keine kontinuierliche Linie gezogen), und Zeile 230 speichert dann die gegenwärtige Blockposition, so daß ein neuer Block berechnet werden kann. Die Schleife wird wiederholt, bis die Raupe den Bildschirmrand erreicht.

Zeile 250 kehrt die Richtung der Raupe um. Die Zeilen 260 und 270 produzieren den "uh-oh"-Ton. Dann bewirkt die Zeile 280, daß die Schleife erneut durchgeführt wird.

Beispiele:

```
>NEW
>100 REM INCHWORM
>110 CALL CLEAR
>120 INPUT "COLOR? ":C
>130 CALL CLEAR
>140 CALL COLOR(2,C,C)
>150 XOLD=1
>160 XDIR=1

>170 FOR I=1 TO 31
>180 XNEW=XOLD+XDIR
>190 CALL HCHAR(12,XNEW,42)
>200 FOR DELAY=1 TO 200
>210 NEXT DELAY
>220 CALL HCHAR(12,XOLD,32)
>230 XOLD=XNEW
>240 NEXT I

>250 XDIR=-XDIR
>260 CALL SOUND(100,392,2)
>270 CALL SOUND(100,330,2)
>280 GOTO 170
>RUN
```

--Bildschirm wird gelöscht

COLOR? 7

--Bildschirm wird gelöscht

--Raupe bewegt sich vor- und rückwärts über den Bildschirm

(Durch CLEAR wird das Programm abgebrochen)

Zeltdach

Dieses Programm bringt ein Zeltdach auf dem Bildschirm.

Die Farbbildung erfolgt zufällig, und jedesmal, wenn ein Farbbalken auf den Bildschirm kommt, ist ein Ton hörbar.

Diese Statements löschen den Bildschirm und ordnen jeden Zeichensatz (2 bis 16) einer anderen Farbe zu. Das RANDOMIZE-Statement stellt sicher, daß bei jedem Programmablauf ein unterschiedlicher Farbensatz erzeugt wird.

Mit diesen Statements wird die Grenze für das Zeltdach gebildet.

Diese Schleife bringt die Farbbalken auf den Bildschirm. Die Bewegung geht von links nach rechts (Spalten 3 bis 30). Bei jeder Anzeige eines Farbbalkens ist ein akustisches Signal hörbar. Die negative Dauer erlaubt, daß der Ton abgeschnitten wird, und daß ein neuer Ton bei jeder Durchführung des CALL SOUND Statements beginnen kann. Das Unterprogramm, das in Zeile 310 beginnt, erzeugt Zufallsfarben und Töne.

Diese Schleife ist die gleiche wie in den Zeilen 200 bis 240, außer daß die Farbbalken in der Folge von rechts nach links auf den Bildschirm gebracht werden. Diese Farbbalken werden unter diejenigen gesetzt, die in der vorangehenden Schleife gebildet wurden. Nach Beendigung der Schleife verzweigt das Programm zur Zeile 200, und beginnt wieder links.

Dieses Unterprogramm erzeugt ein Zufallzeichen (und damit auch eine Zufallsfarbe) für die CALL VCHAR Statements (Zeile 200, 270). Die Zuordnungsanweisungen in den Zeilen 320 und 330 bilden einen Zufallston. Das RETURN-Statement verzweigt das Programm zu den Statements nach GOSUB (Zeile 210, 260).

Beispiele:

```
>NEW

>100 REM MARQUEE
>110 RANDOMIZE
>120 CALL CLEAR
>130 FOR S=2 TO 16
>140 CALL COLOR(S,S,S)
>150 NEXT S

>160 CALL HCHAR(7,3,64,28)
>170 CALL HCHAR(16,3,64,28)
>180 CALL VCHAR(7,2,64,10)
>190 CALL VCHAR(7,31,64,10)

>200 FOR A=3 TO 30
>210 GOSUB 310
>220 CALL VCHAR(8,A,C,4)
>230 CALL SOUND(-150,Y,2)
>240 NEXT A

>250 FOR A=30 TO 3 STEP -1
>260 GOSUB 310
>270 CALL VCHAR(12,A,C,4)
>280 CALL SOUND(-150,Y,2)
>290 NEXT A
>300 GOTO 200

>310 C=INT(120*RND)+40
>320 N=INT(24*RND)+1
>330 Y=220*(2+(1/12))*N
>340 RETURN
>RUN

 --Bildschirm wird gelöscht

 --Zeltdach erscheint

 (Durch CLEAR wird das
 Programm abgebrochen)
```

Geheimzahl

Dieses Programm ist ein Spiel. Es geht darum, eine zufällig gewählte Zahl zu erraten, die zwischen 1 und einer oberen Grenze liegt, die Sie eingeben. Für jeden Rateversuch geben Sie zwei Zahlen ein: eine niedrige und eine hohe Schätzung. Der Computer informiert Sie, ob die Geheimzahl unter, über oder zwischen Ihren beiden Eingabewerten liegt. Wenn Sie glauben, die Zahl zu kennen, geben Sie für den niedrigen und den hohen Schätzwert die gleiche Zahl ein.

Das RANDOMIZE-Statement stellt für jeden Programmablauf eine andere Zahlenfolge sicher. MSG1$ und MSG2$ werden in den PRINT-Statements wiederholt verwendet. Das CALL CLEAR Statement löscht den Bildschirm.

Das INPUT-Statement stoppt das Programm und erwartet die Eingabe einer Grenze. Dann wird die Geheimzahl erzeugt und der Bildschirm gelöscht. Mit N wird die Anzahl Ihrer Rateversuche kontrolliert.

Dieses INPUT-Statement nimmt Ihre hohen und niedrigen Schätzwerte auf. Wenn Sie für beide Werte die gleiche Zahl eingeben und die Geheimzahl raten, verzweigt das Programm zur Zeile 300. Ist die Geheimzahl kleiner als Ihr niedriger Schätzwert, verzweigt das Programm zur Zeile 260. Ist die Geheimzahl größer als Ihr hoher Schätzwert, verzweigt das Programm zur Zeile 280. Liegt die Geheimzahl zwischen Ihren beiden Eingabewerten, oder ist sie mit einem Ihrer Werte identisch, wird das Programm fortgesetzt.

Diese Statements drucken eine Nachricht aus, um Sie zu informieren, wo die Geheimzahl im Verhältnis zu Ihren Rateversuchen einzuordnen ist. Dann verzweigt das Programm zur Zeile 180, um einen erneuten Rateversuch zu ermöglichen. Wenn Sie die Geheimzahl gefunden haben, sagt Ihnen der Computer die Anzahl der Versuche.

Beispiele:

```
>NEW

>100 REM SECRET NUMBER
>110 RANDOMIZE
>120 MSG1$="SECRET NUMBER IS"

>130 MSG2$="YOUR TWO NUMBERS"

>140 CALL CLEAR


>150 INPUT "ENTER LIMIT? ":LI
 MIT
>160 SECRET=INT(LIMIT*RND)+1
>170 CALL CLEAR
>180 N=N+1


>190 INPUT "LOW,HIGH GUESSES:
  ":LOW,HIGH
>200 IF LOW<>HIGH THEN 220
>210 IF SECRET=LOW THEN 300
>220 IF SECRET<LOW THEN 260
>230 IF SECRET>HIGH THEN 280



>240 PRINT MSG1$&" BETWEEN":M
 SG2$
>250 GOTO 180
>260 PRINT MSG1$&" LESS THAN"
 :MSG2$
>270 GOTO 180
>280 PRINT MSG1$&" LARGER THA
 N":MSG2$
>290 GOTO 180
>300 PRINT "YOU GUESSED THE S
 ECRET"
>310 PRINT "NUMBER IN ";N;"TR
 IES"
```

Geheimzahl

Diese Statements bieten Ihnen die Wahl, ein neues Spiel zu machen oder das Programm zu unterbrechen. Wenn Sie ein anderes Zeichen als Y eingeben, endet das Programm. Bei einem erneuten Spiel wird der Zähler für die Zahl der Rateversuche auf null gesetzt. Mit der Eingabe von Y verzweigt das Programm zurück zur Zeile 140. Wenn Sie ein anderes Zeichen eingeben, verzweigt das Programm zurück zur Zeile 160, um eine neue Geheimzahl zu erzeugen.

Hier ist ein Muster des Programmablaufs. (Natürlichen werden Ihre Geheimzahlen nicht mit dem hier gezeigten Beispiel identisch sein).

Beispiele:

```
>320 PRINT "WANT TO PLAY AGAI
 N?"
>330 INPUT "ENTER Y OR N: ":A
 $
>340 IF A$<>"Y" THEN 390
>350 N=0
>360 PRINT "WANT TO SET A NEW
  LIMIT?"
>370 INPUT "ENTER Y OR N: ":B
 $
>380 IF B$="Y" THEN 140 ELSE
 160
>390 END
```

```
>RUN

 --Bildschirm wird gelöscht

 ENTER LIMIT? 20

 --Bildschirm wird gelöscht

 LOW,HIGH GUESSES: 1,10
 SECRET NUMBER IS BETWEEN
 YOUR TWO NUMBERS

 LOW,HIGH GUESSES: 1,5
 SECRET NUMBER IS LARGER THAN
 YOUR TWO NUMBERS

 LOW,HIGH GUESSES: 7,7
 YOU GUESSED THE SECRET
 NUMBER IN  3 TRIES
 WANT TO PLAY AGAIN?
 ENTER Y OR N: N

 ** DONE **
```

Ballspiel

Dieses Programm bewegt einen Ball und schlägt ihn vom Bildschirmrand weg. Bei jedem Auftreffen des Balls auf eine Seite erklingt ein Ton, und der Ball wird zurückgestoßen. Das folgende Spezialzeichen wird für die Definition des Balls verwendet.

Blockkodes

		X	X	X	X			3C
	X	X	X	X	X	X		7E
X	X	X	X	X	X	X	X	FF
X	X	X	X	X	X	X	X	FF
X	X	X	X	X	X	X	X	FF
X	X	X	X	X	X	X	X	FF
	X	X	X	X	X	X		7E
		X	X	X	X			3C

Beispiele:

```
>NEW
```

Diese Statements löschen den Bildschirm und definieren das Zeichen 96 als Ball.

```
>100 REM BOUNCING BALL
>110 CALL CLEAR
>120 CALL CHAR(96,"3C7EFFFFFF
 FF7E3C")
```

Diese Statements erlauben die Eingabe einer Farbe für den Ball und den Bildschirm-Hintergrund. (Beachten Sie, daß die Definition der Bildschirmfarbe durch Anwendung von Zeichensatz 1 für den Bildschirmrand endliche Grenzen ergibt). Der Bildschirm wird gelöscht, wenn die Farben eingegeben sind.

```
>130 INPUT "BALL COLOR? ":C
>140 INPUT "SCREEN COLOR? ":S

>150 CALL CLEAR
>160 CALL COLOR(9,C,S)
>170 CALL COLOR(1,S,S)
```

Diese Statements bewirken, daß der Ball in der Mitte des Bildschirms erscheint, und sich dann nach oben und nach rechts bewegt.

```
>180 X=16
>190 Y=12
>200 XDIR=1
>210 YDIR=1
```

Diese Statements berechnen die nächste Ball-Position. Die Bewegungsrichtung des Balls hängt von den jeweiligen Werten von XDIR (+1 bedeutet nach rechts, —1 nach links) und YDIR (+1 bedeutet nach oben, —1 nach unten) ab.

```
>220 X=X+XDIR
>230 Y=Y+YDIR
```

Diese Statements testen, ob die neue Ballposition noch auf dem Bildschirm ist. Liegt den Wert für die Reihe (Y) oder für die Spalte (X) außerhalb des Bereichs, verzweigt das Programm zu Zeile 310 (Spalte außerhalb des Bereichs), um die Ballrichtung zu ändern.

```
>240 IF X<1 THEN 310
>250 IF X>32 THEN 310
>260 IF Y<1 THEN 360
>270 IF Y>24 THEN 360
```

Ballspiel

Ist die neue Ballposition noch auf dem Bildschirm, wird der Bildschirm gelöscht, um den Ball an der alten Stelle zu entfernen. Der Ball wird dann an der neuen Stelle, die durch Y und X festliegt, angezeigt.

Diese Statements ändern die Ballrichtung, wenn X außerhalb des Bereichs liegt. Das CALL SOUND Statement erzeugt den Ton für das "Aufprallen". In den Zeilen 330 und 340 wird getestet, ob auch Y nicht mehr im Bereich ist. In diesem Fall verzweigt das Programm zur Änderung der Y-Richtung . Wenn Y dagegen noch im Bereich liegt, verzweigt das Programm zu Zeile 220, um eine neue Ballposition zu berechnen.

Diese Statements ändern die Ballrichtung, wenn Y außerhalb des Bereichs liegt. Das CALL SOUND Statement erzeugt den Ton für das "Aufprallen". Das Programm verzweigt dann zu Zeile 220, um eine neue Ballposition zu berechnen.

Beispiele:

```
>280 CALL CLEAR
>290 CALL HCHAR(Y,X,96)
>300 GOTO 220


>310 XDIR=-XDIR
>320 CALL SOUND(30,380,2)
>330 IF Y<1 THEN 360
>340 IF Y>24 THEN 360
>350 GOTO 220


>360 YDIR=-YDIR
>370 CALL SOUND(30,380,2)
>380 GOTO 220
>RUN

 --Bildschirm wird gelöscht

 BALL COLOR? 5
 SCREEN COLOR? 15

 --Ball erscheint auf der Mitte des
   Bildschirm und beginnt zu
   hüpfen.

 (Durch CLEAR wird das
 Programm abgebrochen)
```

Kontostand überprüfen

Einmal im Monat haben alle von uns die Gelegenheit, die ausgegebenen Schecks gegenüber den Bankauszügen zu kontrollieren. Im allgemeinen stimmen der Scheckbestand und Kontostand nicht überein, weil es Schecks und Einzahlungen gibt, die noch nicht berücksichtigt sind. Das Programm verhilft Ihnen schnell zur Bilanz.

Beispiele:

```
>NEW
```

Diese Statements löschen den Bildschirm und erlauben Ihnen die Eingabe des Kontostandes auf Ihrem Bankauszug.

```
>100 REM CHECKBOOK BALANCE
>110 CALL CLEAR
>120 INPUT "BANK BALANCE? ":B
 ALANCE
```

Diese Statements geben Instruktionen für die Eingabe Ihrer noch offenstehenden Schecknummern und Beträge.
Beachten Sie, daß sowohl DISPLAY als auch PRINT verwendet werden kann.

```
>130 DISPLAY "ENTER EACH OUTS
 ANDING"
>140 DISPLAY "CHECK NUMBER AN
 D AMOUNT."
>150 DISPLAY
>160 DISPLAY "ENTER A ZERO FO
 R THE"
>170 DISPLAY "CHECK NUMBER WH
 EN FINISHED."
>180 DISPLAY
```

Diese Schleife stellt das Verfahren für die Eingabe jeder Schecknummer und jedes Betrages auf. Die Werte werden in Datenfeldern gespeichert. Ist die Schecknummer gleich null, verzeigt das Programm aus der Schleife heraus. CTOTAL ist der Gesamtbetrag der ausstehenden Schecks. Bei jeder Eingabe eines Scheckbetrags verzweigt das Programm zu Zeile 190, um eine andere Schecknummer mit einem anderen Betrag einzugeben.

```
>190 N=N+1
>200 INPUT "CHECK NUMBER? ":C
 NUM(N)
>210 IF CNUM(N)=0 THEN 250
>220 INPUT "CHECK AMOUNT? ":C
 AMT(N)
>230 CTOTAL=CTOTAL+CAMT(N)
>240 GOTO 190
```

Diese Statements geben Instruktionen für die Eingabe der noch nicht ausgeführten Einzahlungen.

```
>250 DISPLAY "ENTER EACH OUTS
 TANDING"
>260 DISPLAY "DEPOSIT AMOUNT.
 "
>270 DISPLAY
>280 DISPLAY "ENTER A ZERO AM
 OUNT"
>290 DISPLAY "WHEN FINISHED."

>300 DISPLAY
```

Mit dieser Schleife wird jeder ausstehende Einzahlungsbetrag abgefragt und dann aufgenommen. Wenn der Einzahlungsbetrag gleich null ist, verzweigt das Programm aus der Schleife heraus. DTOTAL ist der Gesamtbetrag der ausstehenden Einzahlungen. Nach der Addition jeder ausstehenden Einzahlung zur Gesamtsumme verzweigt das Programm zur Zeile 310, um einen anderen Einzahlungsbetrag aufzunehmen.

```
>310 M=M+1
>320 INPUT "DEPOSIT AMOUNT? "
 :DAMT(M)
>330 IF DAMT(M)=0 THEN 360
>340 DTOTAL=DTOTAL+DAMT(M)
>350 GOTO 310
```

Kontostand überprüfen

Diese Statements ermitteln den neuen Stand und zeigen ihn an. Dann tragen Sie den augenblicklichen Bestand in Ihr Scheckbuch ein.

(Achten Sie darauf, daß Sie die Service-Gebühren der Bank abziehen, ehe Sie den momentanen Stand eingeben. Die für die Übereinstimmung von Scheckbuch und Bankauszug nötige Korrektur wird dann berechnet und angezeigt).

Beispiele:

```
>360 NBAL=BALANCE-CTOTAL+DTOT
 AL
>370 DISPLAY "NEW BALANCE= ";
 NBAL
>380 INPUT "CHECKBOOK BALANCE
 ? ":CBAL
>390 DISPLAY "CORRECTION= ";N
 BAL-CBAL
>400 END


>RUN

 --Bildschirm wird gelöscht

 BANK BALANCE? 940.26

 ENTER EACH OUTSTANDING
 CHECK NUMBER AND AMOUNT.

 ENTER A ZERO FOR THE
 CHECK NUMBER WHEN FINISHED.

 CHECK NUMBER? 212
 CHECK AMOUNT? 76.83
 CHECK NUMBER? 213
 CHECK AMOUNT? 122.87
 CHECK NUMBER? 216
 CHECK AMOUNT? 219.50
 CHECK NUMBER? 218
 CHECK AMOUNT? 397.31
 CHECK NUMBER? 219
 CHECK AMOUNT? 231.00
 CHECK NUMBER? 220
 CHECK AMOUNT? 138.25
 CHECK NUMBER? 0
 ENTER EACH OUTSTANDING
 DEPOSIT AMOUNT.

 ENTER A ZERO AMOUNT
 WHEN FINISHED.

 DEPOSIT AMOUNT? 450
 DEPOSIT AMOUNT? 0
 NEW BALANCE=  204.5

 CHECKBOOK BALANCE? 209.15
 CORRECTION= -4.65

 ** DONE **
```

Graphikkombinationen

Dieses Spielprogramm gibt Ihnen ein Beispiel für die Entwicklung spezieller Graphikzeichen für den eigenen Gebrauch. Es werden 6 verschiedene graphische Zeichen definiert. Diese sind: ein Herz, eine Kirsche, eine Glocke, eine Zitrone, ein Diamant und ein Rechteck.

Um das Spiel durchzuführen, müssen Sie nur das Programm ablaufen lassen. Der Computer erzeugt drei Zufallszahlen im Bereich 1 bis 6. Bei jeder Bildung einer Zahl wird das der Zahl entsprechende Bild auf dem Schirm angezeigt. Die Punktewertung hängt davon ab, wie viele Bilder übereinstimmen und in welcher Weise. Wenn die drei Bilder und die Punktewertung angezeigt worden sind, haben Sie die Möglichkeit, erneut zu spielen.

Diese Statements definieren die Farben für jedes Zeichen. Die verwendeten Farben sind:

Graphikzeichen	Farbe
Herz	mittelrot
Kirsche	mittelrot mit dunkelgrünem Stiel
Glocke	hellblau mit schwarzem Griff
Zitrone	dunkelgelb
Diamant	dunkelgrün
Rechteck	dunkelblau

Für alle Bilder wird ein weißer Hintergrund verwendet.
Diese Statements definieren das Herz.

Blockkodes **Blockkodes**

00																00
00																00
1C				X	X	X					X	X	X			38
3E			X	X	X	X	X			X	X	X	X	X		7C
7F		X	X	X	X	X	X	X	X	X	X	X	X	X	X	FE
7F		X	X	X	X	X	X	X	X	X	X	X	X	X	X	FE
7F		X	X	X	X	X	X	X	X	X	X	X	X	X	X	FE
7F		X	X	X	X	X	X	X	X	X	X	X	X	X	X	FE
3F			X	X	X	X	X	X	X	X	X	X	X	X		FC
1F				X	X	X	X	X	X	X	X	X	X			F8
0F					X	X	X	X	X	X	X	X				F0
07						X	X	X	X	X	X					E0
03							X	X	X	X						C0
01								X	X							80
00																00
00																00

Beispiele:

```
>NEW

>100 REM GRAPHICS MATCH
>110 CALL COLOR(9,7,16)
>120 CALL COLOR(10,13,16)
>130 CALL COLOR(11,2,16)
>140 CALL COLOR)12,6,16)
>150 CALL COLOR(13,11,16)
>160 CALL COLOR(14,5,16)
```

```
>170 CALL CHAR(96,"00001C3E7F
 7F7F7F")
>180 CALL CHAR(97,"0000387CFE
 FEFEFE")
>190 CALL CHAR(98,"3F1F0F0703
 01")
>200 CALL CHAR(99,"FCF8F0E0C0
 80")
```

Graphikkombinationen

Beachten Sie, daß in den Zeilen 190 und 200 die letzten vier Nullen ausgelassen sind. Dies ist bei der Eingabe der Zeilen zeitsparend, da der Computer automatisch die restliche Länge des Strings mit Nullen auffüllt.

Diese Statements definieren die Kirsche.

Blockkodes																	Blockkodes
00																	00
00																	00
00														X	X		06
00													X				08
00												X					10
1F				X	X	X	X	X			X						20
3F			X	X	X	X	X	X		X							40
7F		X	X	X	X	X	X	X	X								80
7F		X	X	X	X	X	X	X	X	X	X						E0
7F		X	X	X	X	X	X	X	X	X	X	X					F0
7F		X	X	X	X	X	X	X	X	X	X	X					F0
7F		X	X	X	X	X	X	X	X	X	X	X					F0
3F			X	X	X	X	X	X	X	X	X	X					F0
3F			X	X	X	X	X	X	X	X	X						E0
1F				X	X	X	X	X	X	X							C0
00																	00

Diese Statements definieren die Glocke.

Blockkodes																	Blockkodes
00																	00
00																	00
01								X	X								80
01								X	X								80
01								X	X								80
01								X	X								80
01								X	X								80
01								X	X								80
03							X	X	X	X							C0
07						X	X	X	X	X	X						E0
07						X	X	X	X	X	X						E0
07						X	X	X	X	X	X						E0
07						X	X	X	X	X	X						E0
0F					X	X	X	X	X	X	X	X					E0
07						X	X	X	X	X	X						E0
01								X	X								80

Beispiele:

```
>210 CALL CHAR(100,"000000000
 01F3F7F")
>220 CALL CHAR(104,"000006081
 0204080")
>230 CALL CHAR(101,"7F7F7F7F3
 F3F1F")
>240 CALL CHAR(102,"E0F0F0F0F
 0E0C0")
```

```
>250 CALL CHAR(112,"000001010
 1010101")
>260 CALL CHAR(113,"000080808
 0808080")
>270 CALL CHAR(120,"030707070
 70F0701")
>280 CALL CHAR(121,"C0E0E0E0E
 0F0E080")
```

Graphikkombinationen

Diese Statements definieren die Zitrone.

Blockkodes																	Blockkodes
00																	00
00																	00
00																	00
03							X	X	X	X							C0
0F					X	X	X	X	X	X	X	X					F0
1F				X	X	X	X	X	X	X	X	X	X				F8
3F			X	X	X	X	X	X	X	X	X	X	X	X			FC
FF	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	FF
FF	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	FF
3F			X	X	X	X	X	X	X	X	X	X	X	X			FC
1F				X	X	X	X	X	X	X	X	X	X				F8
0F					X	X	X	X	X	X	X	X					F0
03							X	X	X	X							C0
00																	00
00																	00
00																	00

Diese Statements definieren den Diamanten.

Blockkodes																	Blockkodes
00																	00
01								X	X								80
03							X	X	X	X							C0
07						X	X	X	X	X	X						E0
0F					X	X	X	X	X	X	X	X					F0
1F				X	X	X	X	X	X	X	X	X	X				F8
3F			X	X	X	X	X	X	X	X	X	X	X	X			FC
7F		X	X	X	X	X	X	X	X	X	X	X	X	X	X		FE
7F		X	X	X	X	X	X	X	X	X	X	X	X	X	X		FE
3F			X	X	X	X	X	X	X	X	X	X	X	X			FC
1F				X	X	X	X	X	X	X	X	X	X				F8
0F					X	X	X	X	X	X	X	X					F0
07						X	X	X	X	X	X						E0
03							X	X	X	X							C0
01								X	X								80
00																	00

Beispiele:

```
>290 CALL CHAR(128,"000000030
 F1F3FFF")
>300 CALL CHAR(129,"000000C0F
 0F8FCFF")
>310 CALL CHAR(130,"FF3F1F0F0
 3")
>320 CALL CHAR(131,"FFFCF8F0C
 0")
```

```
>330 CALL CHAR(105,"000103070
 F1F3F7F")
>340 CALL CHAR(106,"0080C0E0F
 0F8FCFE")
>350 CALL CHAR(107,"7F3F1F0F0
 70301")
>360 CALL CHAR(108,"FEFCF8F0E
 0C080")
```

Graphikkombinationen

Diese Statements definieren das Rechteck.

Blockkodes																Blockkodes
00																00
00																00
00																00
00																00
00																00
3F			X	X	X	X	X	X	X	X	X	X	X	X		FC
3F			X	X	X	X	X	X	X	X	X	X	X	X		FC
3F			X	X	X	X	X	X	X	X	X	X	X	X		FC
3F			X	X	X	X	X	X	X	X	X	X	X	X		FC
3F			X	X	X	X	X	X	X	X	X	X	X	X		FC
3F			X	X	X	X	X	X	X	X	X	X	X	X		FC
00																00
00																00
00																00
00																00
00																00

Das RANDOMIZE-Statement stellt sicher, daß bei jedem Programmablauf eine andere Bilderfolge erzeugt wird. Die Variable C gibt die Spaltenposition für den Beginn des nächsten Bildes an. Die I-Schleife erzeugt eine Zufallszahl zwischen 1 und 6 inklusive. Das ON-GOSUB-Statement (Zeile 460) verzweigt das Programm zum entsprechenden Unterprogramm, um das Bild auf dem Schirm anzuzeigen. Die Bilder werden nach den folgenden Werten angezeigt.

PIC(I) (Bildnr.)	Bild
1	Herz
2	Kirsche
3	Glocke
4	Zitrone
5	Diamant
6	Rechteck

Nachdem das Bild auf den Schirm gebracht wurde, kehrt das Programm zur Schleife zurück, um eine neue Zahl und ein neues Bild zu erzeugen. Wenn drei Bilder angezeigt sind, wird das Programm mit der Auswertung der Resultate fortgesetzt.

Beispiele:

```
>370 CALL CHAR(136,"000000000
 03F3F3F")
>380 CALL CHAR(137,"000000000
 0FCFCFC")
>390 CALL CHAR(138,"3F3F3F")
>400 CALL CHAR(139,"FCFCFC")
```

```
>410 RANDOMIZE
>420 CALL CLEAR
>430 C=14
>440 FOR I=1 TO 3
>450 PIC(I)=INT(6*RND)+1
>460 ON PIC(I) GOSUB 840,900,
 960,1020,1080,1140
>470 C=C+2
>480 NEXT I
```

Graphikkombinationen

Diese Statements bestimmen die Punktewertung, die Sie erhalten, und die in der nachstehenden Tabelle angegeben ist. Die Zeilennummer verweist auf die Zeile, zu der das Programm verzweigt, um die Punkte gutzuschreiben.

Spiel	Punkte	Zeilennr.
Alle Bilder sind gleich	Gewinn 75	700
Die ersten beiden Bilder sind Kirsche, Zitrone oder Rechteck	Gewinn 40	550
Die ersten beiden Bilder sind Herz, Glocke oder Diamant	Gewinn 10	650
Das erste und das letzte Bild sind gleich	Gewinn 10	650
Keine Übereinstimmung oder Gleichheit der letzten beiden Bilder	Verlust 10	610

Beispiele:

```
>490 REM SCORING
>500 IF PIC(1)<>PIC(2) THEN 5
 20
>510 IF PIC(2)=PIC(3) THEN 70
 0 ELSE 540
>520 IF PIC(1)<>PIC(3) THEN 6
 10
>530 GOTO 650
>540 IF PIC(1)/2<>INT(PIC(1)/
 2) THEN 650
```

Diese Statements addieren 40 Punkte zur akkumulierten Punktezahl. Drei Töne erklingen, und eine Nachricht wird auf dem Bildschirm angezeigt, um anzugeben, daß Sie einen Bonus von 40 Punkten gewonnen haben. Zur Anzeige der gesamten Punktezahl verzweigt dann das Programm zur Zeile 770.

```
>550 TOTAL=TOTAL+40
>560 CALL SOUND(100,440,2)
>570 CALL SOUND(100,660,2)
>580 CALL SOUND(100,550,2)
>590 PRINT "BONUS--40 POINTS"

>600 GOTO 770
```

In Zeile 610 werden 10 Punkte von der Gesamtpunktezahl abgezogen. Ein Ton ist zu hören und eine Nachricht wird angezeigt, um Sie zu informieren, daß Sie 10 Punkte verloren haben. Dann verzweigt das Programm zur Zeile 770, um die neue Punktezahl anzuzeigen.

```
>610 TOTAL=TOTAL-10
>620 CALL SOUND(100,110,1)
>630 PRINT "LOSE 10 POINTS"
>640 GOTO 770
```

In diesen Statements werden zur Gesamtpunktezahl 10 Punkte addiert. Zur Information, daß Sie zehn Punkte gewonnen haben, erklingen zwei Töne, und eine Nachricht wird angezeigt. Zur Anzeige der neuen Punktezahl verzweigt das Programm anschließend zu Zeile 770.

```
>650 TOTAL=TOTAL+10
>660 CALL SOUND(100,660,2)
>670 CALL SOUND(100,770,2)
>680 PRINT "WIN 10 POINTS"
>690 GOTO 770
```

Diese Statements addieren 75 Punkte zur Gesamtpunktezahl. In diesem Fall hören Sie 5 Töne und eine Nachricht wird angezeigt, daß Sie den Hauptgewinn erzielt haben.

```
>700 TOTAL=TOTAL+75
>710 CALL SOUND(100,440,2)
>720 CALL SOUND(100,550,2)
>730 CALL SOUND(100,440,2)
>740 CALL SOUND(100,660,2)
>750 CALL SOUND(100,880,2)
>760 PRINT "JACKPOT!--75 POIN
 TS"
```

Das PRINT-Statement in Zeile 770 druckt Ihre momentane Punktezahl aus. Die anderen Statements bieten Ihnen die Alternative, ob Sie erneut spielen oder das Programm unterbrechen wollen. Das CALL KEY-Statement (Zeile 800) akzeptiert eine Antwort, ohne daß Sie ENTER drücken müssen. Das Drücken der Y-Taste weist das Programm an, zur Erzeugung von drei neuen Bildern zurück zu Zeile 410 zu verzweigen. Wenn Sie eine andere Taste drücken, stoppt das Programm.

```
>770 PRINT "CURRENT TOTAL POI
 NTS: ";TOTAL
>780 PRINT "WANT TO PLAY AGAI
 N?"
>790 PRINT "PRESS Y FOR YES"
>800 CALL KEY(0,KEY,STATUS)
>810 IF STATUS=0 THEN 800
>820 IF KEY=89 THEN 410
>830 END
```

Graphikkombinationen

Diese sechs Unterprogramme drucken jedes der 6 Bilder. Die RETURN-Statements werden so eingesetzt, daß nur ein Bild für jeden Unterprogramm-Aufruf gedruckt wird.

Beispiele:

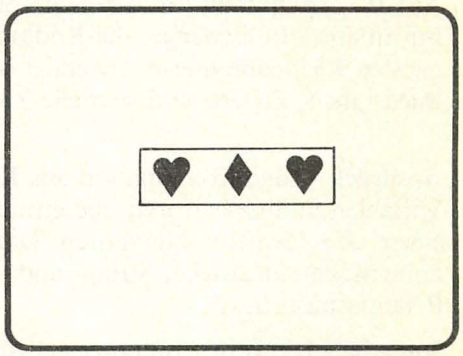
```
>840 REM PRINT HEART
>850 CALL HCHAR(12,C,96)
>860 CALL HCHAR(12,C+1,97)
>870 CALL HCHAR(13,C,98)
>880 CALL HCHAR(13,C+1,99)
>890 RETURN
>900 REM PRINT CHERRY
>910 CALL HCHAR(12,C,100)
>920 CALL HCHAR(12,C+1,104)
>930 CALL HCHAR(13,C,101)
>940 CALL HCHAR(13,C+1,102)
>950 RETURN
>960 REM PRINT BELL
>970 CALL HCHAR(12,C,112)
>980 CALL HCHAR(12,C+1,113)
>990 CALL HCHAR(13,C,120)
>1000 CALL HCHAR(13,C+1,121)
>1010 RETURN
>1020 REM PRINT LEMON
>1030 CALL HCHAR(12,C,128)
>1040 CALL HCHAR(12,C+1,129)
>1050 CALL HCHAR(13,C,130)
>1060 CALL HCHAR(13,C+1,131)
>1070 RETURN
>1080 REM PRINT DIAMOND
>1090 CALL HCHAR(12,C,105)
>1100 CALL HCHAR(12,C+1,106)
>1110 CALL HCHAR(13,C,107)
>1120 CALL HCHAR(13,C+1,108)
>1130 RETURN
>1140 REM PRINT BAR
>1150 CALL HCHAR(12,C,136)
>1160 CALL HCHAR(12,C+1,137)
>1170 CALL HCHAR(13,C,138)
>1180 CALL HCHAR(13,C+1,139)
>1190 RETURN
```

Graphikkombinationen

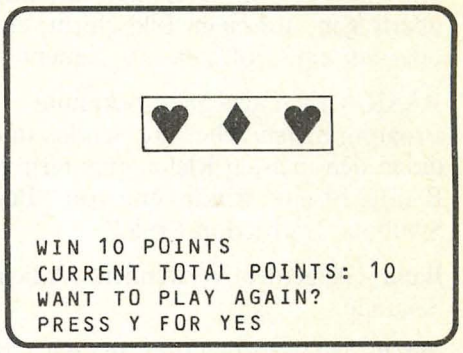
Hier ist der Ablauf eines Musterprogramms. Beachten Sie, daß der Computer-Bildschirm kornblumenblau bleibt, während der Computer die Symboltafel erzeugt und das Programm auf Fehler abtastet. Dieser Vorgang dauert etwa eine Minute.

Beispiele:

```
>RUN

--Bildschirm wird gelöscht
```



```
--2 Tonklang
```




```
PRESS Y FOR YES    N

 ** DONE **
```

Begriffe und Erklärungen

Anweisung - (siehe Statement)

Anzeige(n) - Der Home Computer Bildschirm (Substantiv); der Vorgang, bei dem die Zeichen auf dem Bildschirm erscheinen (Verb);

ASCII - Abkürzung für American Standard Code for Information Interchange, die Kodestruktur, die in den meisten Kleincomputern verwendet wird, um Buchstaben, Ziffern und spezielle Zeichen darzustellen.

Ausdruck - Eine Kombination aus Konstanten, Variablen und Operatoren, die ermittelt werden und zu einem einzelnen Resultat führen. Dazu gehören numerische Ausdrücke, String- und Relationsausdrücke.

Ausgabe - Die vom Computer gelieferte Information

Ausgeben - Die Methode, die Informationen vom Speicher des Computers auf ein anderes Gerät zu übertragen, auf einen Bildschirm, einen Zeilendrucker oder auf ein Großspeicher-Element.

BASIC - Eine allgemein bekannte Programmiersprache, problemlos in der Anwendung, die in den meisten Kleincomputern verwendet wird. BASIC ist eine Abkürzung von "Beginners Allpurpose Symbolic Instruction Code".

Baud - Allgemein verwendete Maßeinheit für Bits pro Sekunde.

Befehl - Eine Instruktion, die der Computer sofort durchführt. Befehle sind nicht Bestandteil eines Programms und werden also ohne vorangehende Zeilennummer eingegeben.

Befehlsmodus - Wenn gerade kein Programm abläuft, befindet sich der Compute im Befehlsmodus und führt jede Aufgabe durch, die Sie eingeben.

Binär - Ein Zahlensystem, das nur auf zwei Ziffern, 0 und 1, basiert. Die interne Sprache und die Operationen des Computers basieren auf dem Binärsystem.

Byte - Eine Folge von aneinandergereihten binären Ziffern (bits), die als Einheit behandelt werden, und oft ein Datenzeichen darstellen. Die Speicherkapazität eines Computers wird als die Anzahl der verfügbaren Bytes ausgedrückt. Ein Computer mit 16K Byte hat zum Beispiel für die Speicherung von Programmen und Daten 16000 Bytes verfügbar.

Programm-Module – Vorprogrammierte ROM-Module, die leicht in das TI-Home Computer System eingeschoben werden können, um seine Fähigkeiten zu erweitern.

Datei - Eine Sammlung von zusammenhängenden Datensätzen, die auf einem anderen Gerät gespeichert sind; auch im gegenseitigen Austausch mit einem Gerät für Ein-/Ausgabe-Elemente verwendet, das keine Mehrfachdateien verwenden kann, zum Beispiel ein Zeilendrucker.

Dateiende - Die Bedingung, die angibt, daß alle Daten aus einer Datei gelesen wurden.

Daten - Grundelemente der Informationen, die vom Computer verarbeitet oder erzeugt werden.

Datenfeld - Eine Sammlung von numerischen Variablen oder String-Variablen, die in einer Liste oder Matrix für die Verarbeitung durch den Computer angeordnet sind. Jedes Element in einem Datenfeld wird durch einen Index bezeichnet, der seine Position in der Liste angibt.

Datenformat Internal - Daten in der Form, wie sie vom Computer direkt verwendet werden. Interne numerische Daten haben ein Länge von 8 Bytes plus 1 Byte für die Längenspezifikation. Die Länge für die internen Stringdaten beträgt 1 Byte pro Zeichen im String plus 1 Längenbyte.

Datensatz - Eine Sammlung von zusammenhängenden Daten, zum Beispiel Informationen zur Gehaltsabrechnung oder die Testergebnisse eines Studenten. Eine Gruppe von gleichartigen Datensätzen, zum Beispiel alle Gehaltsaufzeichnungen eines Unternehmens, werden als Datei bezeichnet.

Datensätze mit fester Länge (FIXED LENGTH) - Datensätze oder Aufzeichnungen in einer Datei, die jeweils die gleiche Länge aufweisen. Hat eine Datei Datensätze mit einer festen Länge von 95 Zeichen, werden jedem Satz 95 Bytes zugeordnet, selbst wenn die Daten nur 76 Stellen belegen. Der Computer addiert am rechten Ende Füllzeichen, um die spezifizierte Länge des Datensatzes zu gewährleisten.

Datensätze mit variabler Länge – Datensätze in einer Datei, die je nach Datenmenge in ihrer Länge variieren. Mit der Verwendung von Sätzen mit variabler Länge spart man Platz auf der Datei. Bei diesen Datensätzen ist nur der sequentielle Zugriff möglich.

Dialogsymbol - a) ein Symbol (>), das den Beginn jedes Befehls oder jeder Programmzeile, die Sie eingeben, markiert;
b) ein Symbol oder ein Satz, mit dem eine Eingabe vom Anwender gefordert wird.

Druckzeile - Eine 28 Positionen umfassende Zeile, die bei den Statements PRINT und DISPLAY verwendet wird.

Begriffe und Erklärungen

Durchführung - der Ablauf eines Programms; die Durchführung der im Statement oder Befehl spezifizierten Aufgabe.

Edit-Modus – Ein Modus, der für die Änderung von existierenden Programmzeilen verwendet wird. Der EDIT-Modus wird mit dem EDIT-Befehl eingegeben. Die spezifizierte Zeile wird auf dem Bildschirm angezeigt, und Sie können mit den unter EDIT beschriebenen speziellen Tasten jedes beliebige Zeichen ändern.

Eingabe - Daten, die in den Speicher des Computers zu plazieren sind.

Eingabezeile - Die Datenmenge, die auf einmal eingegeben werden kann. In TI-BASIC sind dies 112 Zeichen.

Eingeben - Das Verfahren, Daten in den Speicher zu übertragen.

Exponent - eine Zahl, die die Potenz angibt, in die eine Zahl oder ein Ausdruck erhoben wird; gewöhnlich wird der Exponent rechts oberhalb der Zahl geschrieben. Zum Beispiel 2^8 = $2\times2\times2\times2\times2\times2\times2\times2$. In TI-BASIC wird der Exponent nach dem Symbol $\wedge$ oder nach dem Buchstaben "E" in der der Exponentialform eingegeben. Zum Beispiel:
$2^8 = 2 \wedge 8$; $1.3 \times 10^{25} = 1.3E25$

Exponentialform - Eine Methode, sehr große oder sehr kleine Zahlen auszudrücken: eine Basiszahl (Mantisse) wird mit einer Zehnerpotenz (Exponent) multipliziert. Zur Darstellung der Exponentialform in TI-BASIC geben Sie das Vorzeichen, dann die Mantisse, den Buchstaben E und die Zehnerpotenz ein, (der wiederum das Minuszeichen vorangestellt wird, wenn sie negativ ist).
Beispiel: 3.264E4; —2.47E—17.

Fest(wert)speicher - siehe ROM

Funktion - Ein Merkmal, mit dem es möglich ist, eine Vielzahl von Vorgängen als Einzel-Operationen zu spezifizieren, die aber tatsächlich eine Reihe von Schritten enthalten; zum Beispiel ein Verfahren zur Berechnung der Quadratwurzel über ein einfaches Bezugswort.

Ganze Zahl - Eine positive oder negative ganze Zahl oder null.

Graphik - Visuelle Abbildungen auf dem Schirm, wie graphische Darstellungen, Muster, und Zeichnungen, in bewegter und unbewegter Form. TI-BASIC verfügt über eingebaute Unterprogramme, die leicht anwendbare Möglichkeiten mit Farbgraphiken eröffnen.

Graphikzeile - Eine 32 Zeichen umfassende Zeile, die bei den Graphik-Unterprogrammen in TI-BASIC verwendet wird.

Großspeicher-Element – Ein Zubehörgerät, zum Beispiel ein Kassettenrekorder oder ein Diskettensystem, das Programme und/oder Daten für die spätere Verwendung durch den Computer speichert. Diese Informationen werden im allgemeinen in einem Format aufgezeichnet, das vom Computer, nicht aber vom Menschen gelesen werden kann.

Hardware – Die verschiedenen Geräte, die zusammen das Computersystem bilden, einschließlich Speicher, Tastatur, Bildschirm, Diskettensysteme, Zeilendrucker etc.

Hertz (Hz) - Maßeinheit für die Frequenz; 1 Hertz = 1 Schwingung pro Sekunde.

Hexadezimal - Ein Zahlensystem mit der Basis 16, bei dem 16 Symbole, 0 bis 9 und A bis F, verwendet werden. Es wird als vorteilhafte Kurzmethode eingesetzt, um den Binärkode auszudrücken. 1010 in Binärschreibweise ist zum Beispiel A in der Hexadezimalschreibweise, 11111111 entspricht FF. Das Hexadezimalsystem wird bei der Bildung von Mustern für die Graphikzeichen im CALL CHAR Unterprogramm verwendet.

Index - Ein numerischer Ausdruck, der ein bestimmtes Element in einem Datenfeld spezifiziert. In TI-BASIC wird unmittelbar nach der Bezeichnung für das Datenfeld der Index in Klammern geschrieben.

Inkrement - Ein positiver oder negativer Wert, der eine Variable kontinuierlich modifiziert.

I/O (Input/Output) – Bezieht sich im allgemeinen auf die Funktion eines Geräts; I/O (Ein-/Ausgabe) wird für die Kommunikation zwischen dem Computer und anderen Geräten oder Elementen verwendet (zum Beispiel Tastatur, Diskette).

Iteration - Die Technik, eine Gruppe von Programm-Statements zu wiederholen; die einzelne Wiederholung von einer derartigen Gruppe; siehe Schleife.

Kapazitätsüberlauf - Die Situation, die dann eintritt, wenn ein gerundeter Wert größer als 9.9999999999999E127 oder kleiner als —9.9999999999999E127 eingegeben oder berechnet wird. In diesem Fall wird der Wert durch die Grenze der Computerkapazität ersetzt, eine Warnung angezeigt und das Programm fortgesetzt.

Begriffe und Erklärungen

Konstante - Ein spezifischer numerischer Wert oder String-Wert. Eine numerische Konstante ist jede beliebige reelle Zahl, zum Beispiel 1.2 oder —9054. Eine String-Konstante ist jede beliebige Kombination von bis zu 112 Zeichen, die in Anführungszeichen gesetzt werden, zum Beispiel "HELLO THERE" oder "275 FIRST ST".

Mantisse - Der Basisteil einer Zahl, die in der Exponentialform ausgedrückt ist. Im Beispiel 3.264E + 4 ist die Mantisse 3.264.

Modul – Siehe Programm-Modul.

Null-String - Ein String, der keine Zeichen enthält, und die Länge null hat.

Number-Modus - Ein Modus, den der Computer annimmt, wenn die Nummern der Programmzeilen für die Eingabe oder Änderung von Statements automatisch erzeugt werden.

Operator - Ein Symbol, das für Berechnungen oder für Vergleiche verwendet wird (numerische Operatoren und Vergleichsoperatoren). Die numerischen Operatoren sind +, —, *, /, ∧ .
Die Vergleichsoperatoren sind
>, <, =, > =, < =, < >.

Parameter - Jede beliebige Wertegruppe, die die Ausgabe eines Statements oder einer Funktion bestimmt oder beeinflußt.

Diskette (Platte) – Großspeicher mit der Möglichkeit des direkten und des sequentiellen Zugriffs.

Positionsanzeiger - Ein Symbol, das angibt, wo das nächste Zeichen auf dem Bildschirm erscheint, wenn Sie eine Taste drücken.

Programm - Eine Reihe von Statements (Anweisungen), die den Computer anweisen, wie eine vollständige Aufgabe durchzuführen ist.

Programm-Unterbrechung - (siehe Stoppstelle).

Programmzeile - Eine Zeile, die ein Einzel-Statement enthält. Die Maximallänge einer Programmzeile ist auf 112 Zeichen limitiert.

Pseudozufallszahl - Eine Zahl, die durch eine bestimmte Reihe von Berechnungen (Algorithmus) erzeugt wird, die aber dennoch den Anforderungen für Zufallszahlen genügt, um für spezielle Zwecke als solche betrachtet zu werden.
Eine echte Zufallszahl wird völlig auf Zufallsbasis gewonnen.

Pufferspeicher - Ein Bereich des Computerspeichers für die temporäre Speicherung eines Ein- oder Ausgabe-Datensatzes.

Radix-100 - Eine Zahlenschreibweise, basierend auf 100.

RAM (Random Access Memory) - Der Hauptspeicher, wo die Programm-Statements und die Daten temporär während der Programmablaufs gespeichert werden. Neue Programme und Daten können eingelesen, aufgerufen und im RAM geändert werden. Die im RAM gespeicherten Daten werden gelöscht, wenn das Gerät ausgeschaltet wird, oder wenn man den BASIC-Modus verläßt.

Rauschen - Verschiedene akustische Signale, mit denen man interessante Klangeffekte erzeugen kann. Ein Rauschen wird im Gegensatz zu einem Ton dann erzeugt, wenn ein negativer Frequenzwert (—1 bis —8) angegeben wird. Das entsprechende Unterprogramm in TI-BASIC ist CALL SOUND.

Reserviertes Wort - Ein Spezialwort in Programmiersprachen mit einer bereits definierten Bedeutung. Ein reserviertes Wort muß orthographisch korrekt in der richtigen Reihenfolge in einem Statement oder in einem Befehl erscheinen und kann nicht als Variablenbezeichnung verwendet werden.

ROM (Read Only Memory) - Bestimmte Instruktionen für den Computer werden permanent im ROM gespeichert; der Zugriff ist möglich, nicht aber Veränderungen. Beim Ausschalten des Geräts wird das ROM nicht gelöscht.

Run-Modus – Wenn der Computer ein Programm durchführt, befindet er sich im RUN-Modus. Der RUN-Modus endet, wenn der Programmablauf tatsächlich oder aufgrund einer Fehlerbedingung endet. Mit CLEAR kann man den RUN-Modus verlassen (siehe Stoppstelle).

Schleife - Eine Gruppe von aufeinanderfolgenden Programmzeilen, die wiederholt durchgeführt werden. Gewöhnlich ist die Anzahl der Wiederholungen spezifiziert.

Software - Verschiedene Programme, die der Computer durchführt, inklusive die in den Computer eingebauten Programme, Programme der Command Module, und Programme, die vom Anwender eingegeben werden.

Speicher - siehe RAM, ROM und Großspeicher-Element.

Sprung - (siehe Verzweigung).

Begriffe und Erklärungen

Standardattribut - Eine Eigenschaft oder ein Wert, den der Computer automatisch voraussetzt, wenn innerhalb eines Statements oder eines Programms bestimmte Spezifikationen ausgelassen werden.

Statement - Eine Anweisung, der eine Zeilennummer im Programm vorangestellt ist. In TI-BASIC ist nur ein Statement in jeder Programmzeile zulässig.

Stoppstelle - Ein durch den BREAK-Befehl spezifizierter Punkt im Programm, wo der Programmablauf unterbrochen werden kann. Während einer Programm-Unterbrechung oder Stoppstelle können Sie Operationen im Befehlsmodus durchführen, um Programmfehler besser lokalisieren zu können. Der Programmablauf wird mit dem CONTINUE-Befehl wieder aufgenommen, wenn nicht während der Unterbrechung editiert wurde.

String - Eine Reihe von Buchstaben, Zahlen und Symbolen, die als Einheit behandelt werden.

Subroutine - Ein Programmsegment, das mehr als einmal während des Programmablaufs verwendet werden kann, zum Beispiel ein komplexer Satz von Berechnungen oder eine Druckroutine. In TI-BASIC wird dieses Programmelement über das Statement GOSUB eingegeben und endet mit dem RETURN-Statement.

TRACE - Auflistung der Reihenfolge, in der der Computer die Programm-Statements durchführt. Die Verfolgung der Zeilennummern mit TRACE kann Ihnen helfen, Fehler im Programmfluß zu finden.

Unterlauf - Die Bedingung, die dann eintritt, wenn der Computer einen numerischen Wert bildet, der größer als —1E—128, kleiner als 1E—128 und nicht null ist. In diesem Fall wird der Wert durch null ersetzt.

Unterprogramm (Teilprogramm) - Ein prädefiniertes, allgemeines Verfahren, das in TI-BASIC über das Statement CALL dem Anwender zugänglich ist. Teilprogramme erweitern die Möglichkeiten des BASIC und können nicht ohne Schwierigkeiten in BASIC programmiert werden.

Variable - Die Bezeichnung für einen Wert, der während der Programmdurchführung variieren kann. Man kann sich die Variable als einen Speicherplatz vorstellen, an dem während des Programmablaufs die Werte durch neue Werte ersetzt werden können.

Verkettung - Verbindung von zwei oder mehr Strings, um einen längeren String zu bilden. Das Zeichen "&" ist der Verkettungs-Operator.

Verzweigung - Eine Abweichung von der sequentiellen Durchführung der Programm-Statements. Eine unbedingte Verzweigung bewirkt, daß der Computer jedesmal zu der spezifizierten Programmzeile springt, wenn eine Verzweigungsanweisung erreicht wird. Bei einer bedingten Verzweigung basiert die Steuerung des Programmablaufs auf dem Resultat einer arithmetischen oder logischen Operation.

Zeichen - Ein Buchstabe, eine Ziffer, ein Interpunktionssymbol oder ein anderes spezielles graphisches Zeichen.

Zeile - Siehe Graphikzeile, Eingabezeile, Druckzeile oder Programmzeile.

Zeitweises Versagen - Ein Hardware-Defekt oder ein Programmierfehler, der zur Folge hat, daß die beabsichtigte Operation nicht korrekt durchgeführt wird.

Zubehör – Zusätzliche Geräte, die an den Computer angeschlossen werden, und die seine Funktionen und Möglichkeiten noch erweitern. Dazu gehören auch die vorprogrammierten Programm-Module und Geräte, die Computerdaten senden, empfangen oder speichern, wie Drucker und Diskettensysteme. Man spricht in diesem Zusammenhang auch von Peripheriegeräten.

TEXAS INSTRUMENTS

1103000-0002

DNOm910C419HC